Package: plotly.core (via r-universe)

November 17, 2026

Type Package

Title Create Interactive Web Graphics via 'plotly.js'

Version 0.1.0

Description Create interactive web graphics with the 'plotly.js' JavaScript library. Graph objects, their validation and their documentation are generated from the 'plotly.js' plot schema, following the design of the 'plotly' Python package. Includes an 'express' interface for quick figures, 'htmlwidgets' rendering, 'shiny' and 'quarto' integration, and static image export with headless 'Chrome'.

License MIT + file LICENSE

Copyright See file inst/COPYRIGHTS

Depends R (>= 4.1.0)

Imports base64enc, data.table, grDevices, htmltools, htmlwidgets, jsonlite, stats, utils

Suggests callr, chromote (>= 0.5.0), crosstalk, httpuv, knitr, listviewer, plotly, png, processx, promises, ps, quarto, rmarkdown, rsvg, shiny

VignetteBuilder quarto

URL https://msummersgill.github.io/plotly.R.core/, https://github.com/msummersgill/plotly.R.core

BugReports https://github.com/msummersgill/plotly.R.core/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/pak/sysreqs cmake make libuv1-dev

Repository https://msummersgill.r-universe.dev

Date/Publication 2026-11-17 23:18:49 UTC

RemoteUrl https://github.com/msummersgill/plotly.R.core

RemoteRef HEAD

RemoteSha af19892d5024ea2f8d8d8d7d6713a237fd7af7b0

Contents

add_annotation . . . 5
add_funnel . . . 14
add_funnelarea . . . 37
add_heatmap . . . 49
add_histogram . . . 68
add_histogram2d . . . 96
add_histogram2dcontour . . . 116
add_icicle . . . 138
add_image . . . 160
add_indicator . . . 168
add_isosurface . . . 183
add_layout_image . . . 204
add_mesh3d . . . 207
add_ohlc . . . 226
add_parcats . . . 237
add_parcoords . . . 252
add_pie . . . 267
add_quiver . . . 281
add_sankey . . . 301
add_scatter . . . 314
add_scatter3d . . . 345
add_scattercarpet . . . 372
add_scattergeo . . . 397
add_scattergl . . . 420
add_scattermap . . . 447
add_scatterpolar . . . 466
add_scatterpolargl . . . 490
add_scattersmith . . . 514
add_scatterternary . . . 538
add_selection . . . 562
add_shape . . . 565
add_splom . . . 574
add_streamtube . . . 595
add_sunburst . . . 613
add_surface . . . 633
add_table . . . 654
add_traces . . . 664
add_treemap . . . 665
add_violin . . . 687
add_volume . . . 703
add_waterfall . . . 724
as_float32 . . . 740
chrome_exporter . . . 741
config . . . 742
event_data . . . 743
event_register . . . 744

get_template . . . 745
go_bar . . . 746
go_barpolar . . . 772
go_box . . . 792
go_candlestick . . . 809
go_carpet . . . 819
go_choropleth . . . 834
go_choroplethmap . . . 851
go_cone . . . 867
go_contour . . . 885
go_contourcarpet . . . 906
go_densitymap . . . 922
go_figure . . . 937
go_layout . . . 938
go_layout_activeselection . . . 947
go_layout_activeshape . . . 948
go_layout_coloraxis . . . 949
go_layout_colorscale . . . 957
go_layout_font . . . 959
go_layout_geo . . . 960
go_layout_grid . . . 966
go_layout_hoverlabel . . . 969
go_layout_legend . . . 971
go_layout_map . . . 977
go_layout_margin . . . 984
go_layout_modebar . . . 985
go_layout_newselection . . . 986
go_layout_newshape . . . 988
go_layout_polar . . . 993
go_layout_scene . . . 1006
go_layout_slider . . . 1028
go_layout_smith . . . 1034
go_layout_ternary . . . 1039
go_layout_title . . . 1053
go_layout_transition . . . 1056
go_layout_uniformtext . . . 1058
go_layout_updatemenu . . . 1059
go_layout_xaxis . . . 1063
go_layout_yaxis . . . 1081
highlight . . . 1097
install_chrome . . . 1099
last_plot . . . 1100
make_subplots . . . 1100
partial_bundle . . . 1102
plotly_json . . . 1104
plotly_obj-modify . . . 1104
plotly_widget . . . 1105
plotlyjs_version . . . 1106

plotlyOutput 1107
plotlyProxy 1108
print.go_figure 1109
print.plotly_obj 1110
print_grid 1110
px_area 1111
px_bar 1115
px_bar_polar 1119
px_box 1123
px_choropleth 1126
px_choropleth_map 1130
px_colors 1133
px_constant 1134
px_data 1135
px_defaults 1136
px_density_contour 1137
px_density_heatmap 1141
px_density_map 1145
px_ecdf 1147
px_funnel 1152
px_funnel_area 1155
px_get_trendline_results 1157
px_histogram 1158
px_icicle 1162
px_imshow 1164
px_line 1167
px_line_3d 1171
px_line_geo 1175
px_line_map 1180
px_line_polar 1183
px_line_ternary 1186
px_parallel_categories 1190
px_parallel_coordinates 1191
px_pie 1193
px_scatter 1196
px_scatter_3d 1201
px_scatter_geo 1205
px_scatter_map 1209
px_scatter_matrix 1213
px_scatter_polar 1215
px_scatter_ternary 1219
px_strip 1222
px_sunburst 1225
px_timeline 1228
px_treemap 1232
px_violin 1235
schema 1238
select_traces 1239

set_crosstalk_keys 1240
TeX 1241
update_annotations 1242
update_coloraxes 1243
update_geos 1244
update_layout 1245
update_layout_images 1246
update_legends 1247
update_maps 1248
update_polars 1249
update_scenes 1250
update_selections 1251
update_shapes 1252
update_smiths 1254
update_ternaries 1255
update_xaxes 1256
update_yaxes 1257
write_html 1258

Index **1260**

`add_annotation` *Layout annotation objects*

Description

Properties of `layout.annotation`.

Usage

```
add_annotation(
  fig,
  arg = NULL,
  ...,
  align = NULL,
  arrowcolor = NULL,
  arrowhead = NULL,
  arrowside = NULL,
  arrowsize = NULL,
  arrowwidth = NULL,
  ax = NULL,
  axref = NULL,
  ay = NULL,
  ayref = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  borderpad = NULL,
  borderwidth = NULL,
```

```
  captureevents = NULL,
  clicktoshow = NULL,
  font = NULL,
  height = NULL,
  hoverlabel = NULL,
  hovertext = NULL,
  name = NULL,
  opacity = NULL,
  showarrow = NULL,
  standoff = NULL,
  startarrowhead = NULL,
  startarrowsize = NULL,
  startstandoff = NULL,
  templateitemname = NULL,
  text = NULL,
  textangle = NULL,
  valign = NULL,
  visible = NULL,
  width = NULL,
  x = NULL,
  xanchor = NULL,
  xclick = NULL,
  xref = NULL,
  xshift = NULL,
  y = NULL,
  yanchor = NULL,
  yclick = NULL,
  yref = NULL,
  yshift = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  exclude_empty_subplots = FALSE,
  skip_invalid = FALSE
)

go_layout_annotation(
  arg = NULL,
  ...,
  align = NULL,
  arrowcolor = NULL,
  arrowhead = NULL,
  arrowside = NULL,
  arrowsize = NULL,
  arrowwidth = NULL,
  ax = NULL,
  axref = NULL,
  ay = NULL,
```

```
  ayref = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  borderpad = NULL,
  borderwidth = NULL,
  captureevents = NULL,
  clicktoshow = NULL,
  font = NULL,
  height = NULL,
  hoverlabel = NULL,
  hovertext = NULL,
  name = NULL,
  opacity = NULL,
  showarrow = NULL,
  standoff = NULL,
  startarrowhead = NULL,
  startarrowsize = NULL,
  startstandoff = NULL,
  templateitemname = NULL,
  text = NULL,
  textangle = NULL,
  valign = NULL,
  visible = NULL,
  width = NULL,
  x = NULL,
  xanchor = NULL,
  xclick = NULL,
  xref = NULL,
  xshift = NULL,
  y = NULL,
  yanchor = NULL,
  yclick = NULL,
  yref = NULL,
  yshift = NULL,
  skip_invalid = FALSE
)

go_layout_annotation_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
```

```
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_annotation_hoverlabel(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  skip_invalid = FALSE
)

go_layout_annotation_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

`fig`	A `go_figure` object.
`arg`	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
`...`	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
`align`	Sets the horizontal alignment of the 'text' within the box. Has an effect only if 'text' spans two or more lines (i.e. 'text' contains one or more HTML tags) or if an explicit width is set to override the text width. One of `"left"`, `"center"`, `"right"`.
`arrowcolor`	Sets the color of the annotation arrow. Color: a CSS color string or R color name.
`arrowhead`	Sets the end annotation arrow head style. Integer in the interval [0, 8].
`arrowside`	Sets the annotation arrow head position. Any combination of `"end"`, `"start"` joined with `"+"`, or one of `"none"`.
`arrowsize`	Sets the size of the end annotation arrow head, relative to 'arrowwidth'. A value of 1 (default) gives a head about 3x as wide as the line. Number in the interval

[0.3, Inf].

`arrowwidth` Sets the width (in px) of annotation arrow line. Number in the interval [0.1, Inf].

`ax` Sets the x component of the arrow tail about the arrow head. If ‘axref‘ is ‘pixel‘, a positive (negative) component corresponds to an arrow pointing from right to left (left to right). If ‘axref‘ is not ‘pixel‘ and is exactly the same as ‘xref‘, this is an absolute value on that axis, like ‘x‘, specified in the same coordinates as ‘xref‘. Any value.

`axref` Indicates in what coordinates the tail of the annotation (ax,ay) is specified. If set to a x axis id (e.g. "x" or "x2"), the ‘x‘ position refers to a x coordinate. If set to "paper", the ‘x‘ position refers to the distance from the left of the plotting area in normalized coordinates where 0 (1) corresponds to the left (right). If set to a x axis ID followed by "domain" (separated by a space), the position behaves like for "paper", but refers to the distance in fractions of the domain length from the left of the domain of that axis: e.g., "x2 domain" refers to the domain of the second x axis and a x position of 0.5 refers to the point between the left and the right of the domain of the second x axis. In order for absolute positioning of the arrow to work, "axref" must be exactly the same as "xref", otherwise "axref" will revert to "pixel" (explained next). For relative positioning, "axref" can be set to "pixel", in which case the "ax" value is specified in pixels relative to "x". Absolute positioning is useful for trendline annotations which should continue to indicate the correct trend when zoomed. Relative positioning is useful for specifying the text offset for an annotated point. One of `"pixel"`, or a string matching `"^x([2-9]|[1-9][0-9]+)?( domain)?$"`.

`ay` Sets the y component of the arrow tail about the arrow head. If ‘ayref‘ is ‘pixel‘, a positive (negative) component corresponds to an arrow pointing from bottom to top (top to bottom). If ‘ayref‘ is not ‘pixel‘ and is exactly the same as ‘yref‘, this is an absolute value on that axis, like ‘y‘, specified in the same coordinates as ‘yref‘. Any value.

`ayref` Indicates in what coordinates the tail of the annotation (ax,ay) is specified. If set to a y axis id (e.g. "y" or "y2"), the ‘y‘ position refers to a y coordinate. If set to "paper", the ‘y‘ position refers to the distance from the bottom of the plotting area in normalized coordinates where 0 (1) corresponds to the bottom (top). If set to a y axis ID followed by "domain" (separated by a space), the position behaves like for "paper", but refers to the distance in fractions of the domain length from the bottom of the domain of that axis: e.g., "y2 domain" refers to the domain of the second y axis and a y position of 0.5 refers to the point between the bottom and the top of the domain of the second y axis. In order for absolute positioning of the arrow to work, "ayref" must be exactly the same as "yref", otherwise "ayref" will revert to "pixel" (explained next). For relative positioning, "ayref" can be set to "pixel", in which case the "ay" value is specified in pixels relative to "y". Absolute positioning is useful for trendline annotations which should continue to indicate the correct trend when zoomed. Relative positioning is useful for specifying the text offset for an annotated point. One of `"pixel"`, or a string matching `"^y([2-9]|[1-9][0-9]+)?( domain)?$"`.

`bgcolor` `go_layout_annotation()`: Sets the background color of the annotation. Color: a CSS color string or R color name.

	`go_layout_annotation_hoverlabel()`: Sets the background color of the hover label. By default uses the annotation's 'bgcolor' made opaque, or white if it was transparent. Color: a CSS color string or R color name.
`bordercolor`	`go_layout_annotation()`: Sets the color of the border enclosing the annotation 'text'. Color: a CSS color string or R color name. `go_layout_annotation_hoverlabel()`: Sets the border color of the hover label. By default uses either dark grey or white, for maximum contrast with 'hoverlabel.bgcolor'. Color: a CSS color string or R color name.
`borderpad`	Sets the padding (in px) between the 'text' and the enclosing border. Number in the interval [0, Inf].
`borderwidth`	Sets the width (in px) of the border enclosing the annotation 'text'. Number in the interval [0, Inf].
`captureevents`	Determines whether the annotation text box captures mouse move and click events, or allows those events to pass through to data points in the plot that may be behind the annotation. By default 'captureevents' is FALSE unless 'hovertext' is provided. If you use the event 'plotly_clickannotation' without 'hovertext' you must explicitly enable 'captureevents'. Logical.
`clicktoshow`	Makes this annotation respond to clicks on the plot. If you click a data point that exactly matches the 'x' and 'y' values of this annotation, and it is hidden (visible: false), it will appear. In "onoff" mode, you must click the same point again to make it disappear, so if you click multiple points, you can show multiple annotations. In "onout" mode, a click anywhere else in the plot (on another data point or not) will hide this annotation. If you need to show/hide this annotation in response to different 'x' or 'y' values, you can set 'xclick' and/or 'yclick'. This is useful for example to label the side of a bar. To label markers though, 'standoff' is preferred over 'xclick' and 'yclick'. One of `"onoff"`, `"onout"`, `FALSE`.
`font`	`go_layout_annotation()`: Sets the annotation text font. Named list or `go_layout_annotation_font()` object. `go_layout_annotation_hoverlabel()`: Sets the hover label text font. By default uses the global hover font and size, with color from 'hoverlabel.bordercolor'. Named list or `go_layout_annotation_hoverlabel_font()` object.
`height`	Sets an explicit height for the text box. null (default) lets the text set the box height. Taller text will be clipped. Number in the interval [1, Inf].
`hoverlabel`	Named list or `go_layout_annotation_hoverlabel()` object.
`hovertext`	Sets text to appear when hovering over this annotation. If omitted or blank, no hover label will appear. String.
`name`	When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.
`opacity`	Sets the opacity of the annotation (text + arrow). Number in the interval [0, 1].
`showarrow`	Determines whether or not the annotation is drawn with an arrow. If TRUE, 'text' is placed near the arrow's tail. If FALSE, 'text' lines up with the 'x' and 'y' provided. Logical.

`standoff` Sets a distance, in pixels, to move the end arrowhead away from the position it is pointing at, for example to point at the edge of a marker independent of zoom. Note that this shortens the arrow from the ‘ax‘ / ‘ay‘ vector, in contrast to ‘xshift‘ / ‘yshift‘ which moves everything by this amount. Number in the interval [0, Inf].

`startarrowhead` Sets the start annotation arrow head style. Integer in the interval [0, 8].

`startarrowsize` Sets the size of the start annotation arrow head, relative to ‘arrowwidth‘. A value of 1 (default) gives a head about 3x as wide as the line. Number in the interval [0.3, Inf].

`startstandoff` Sets a distance, in pixels, to move the start arrowhead away from the position it is pointing at, for example to point at the edge of a marker independent of zoom. Note that this shortens the arrow from the ‘ax‘ / ‘ay‘ vector, in contrast to ‘xshift‘ / ‘yshift‘ which moves everything by this amount. Number in the interval [0, Inf].

`templateitemname`
Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`text` Sets the text associated with this annotation. Plotly uses a subset of HTML tags to do things like newline (‘
‘), bold (‘‘), italics (‘‘), hyperlinks (‘‘). Tags ‘‘, ‘<sup>‘, ‘<sub>‘, ‘<s>‘, ‘<u>‘, and ‘‘ are also supported. String.

`textangle` Sets the angle at which the ‘text‘ is drawn with respect to the horizontal. Angle in degrees.

`valign` Sets the vertical alignment of the ‘text‘ within the box. Has an effect only if an explicit height is set to override the text height. One of `"top"`, `"middle"`, `"bottom"`.

`visible` Determines whether or not this annotation is visible. Logical.

`width` Sets an explicit width for the text box. null (default) lets the text set the box width. Wider text will be clipped. There is no automatic wrapping; use <br> to start a new line. Number in the interval [1, Inf].

`x` Sets the annotation’s x position. If the axis ‘type‘ is "log", then you must take the log of your desired range. If the axis ‘type‘ is "date", it should be date strings, like date data, though Date objects and unix milliseconds will be accepted and converted to strings. If the axis ‘type‘ is "category", it should be numbers, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`xanchor` Sets the text box’s horizontal position anchor This anchor binds the ‘x‘ position to the "left", "center" or "right" of the annotation. For example, if ‘x‘ is set to 1, ‘xref‘ to "paper" and ‘xanchor‘ to "right" then the right-most portion of the annotation lines up with the right-most edge of the plotting area. If "auto", the anchor is equivalent to "center" for data-referenced annotations or if there is an arrow,

whereas for paper-referenced with no arrow, the anchor picked corresponds to the closest side. One of `"auto"`, `"left"`, `"center"`, `"right"`.

`xclick` Toggle this annotation when clicking a data point whose 'x' value is 'xclick' rather than the annotation's 'x' value. Any value.

`xref` Sets the annotation's x coordinate axis. If set to a x axis id (e.g. "x" or "x2"), the 'x' position refers to a x coordinate. If set to "paper", the 'x' position refers to the distance from the left of the plotting area in normalized coordinates where 0 (1) corresponds to the left (right). If set to a x axis ID followed by "domain" (separated by a space), the position behaves like for "paper", but refers to the distance in fractions of the domain length from the left of the domain of that axis: e.g., "x2 domain" refers to the domain of the second x axis and a x position of 0.5 refers to the point between the left and the right of the domain of the second x axis. One of `"paper"`, or a string matching `"^x([2-9]|[1-9][0-9]+)?(domain)?$"`.

`xshift` Shifts the position of the whole annotation and arrow to the right (positive) or left (negative) by this many pixels. Number.

`y` Sets the annotation's y position. If the axis 'type' is "log", then you must take the log of your desired range. If the axis 'type' is "date", it should be date strings, like date data, though Date objects and unix milliseconds will be accepted and converted to strings. If the axis 'type' is "category", it should be numbers, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`yanchor` Sets the text box's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the annotation. For example, if 'y' is set to 1, 'yref' to "paper" and 'yanchor' to "top" then the top-most portion of the annotation lines up with the top-most edge of the plotting area. If "auto", the anchor is equivalent to "middle" for data-referenced annotations or if there is an arrow, whereas for paper-referenced with no arrow, the anchor picked corresponds to the closest side. One of `"auto"`, `"top"`, `"middle"`, `"bottom"`.

`yclick` Toggle this annotation when clicking a data point whose 'y' value is 'yclick' rather than the annotation's 'y' value. Any value.

`yref` Sets the annotation's y coordinate axis. If set to a y axis id (e.g. "y" or "y2"), the 'y' position refers to a y coordinate. If set to "paper", the 'y' position refers to the distance from the bottom of the plotting area in normalized coordinates where 0 (1) corresponds to the bottom (top). If set to a y axis ID followed by "domain" (separated by a space), the position behaves like for "paper", but refers to the distance in fractions of the domain length from the bottom of the domain of that axis: e.g., "y2 domain" refers to the domain of the second y axis and a y position of 0.5 refers to the point between the bottom and the top of the domain of the second y axis. One of `"paper"`, or a string matching `"^y([2-9]|[1-9][0-9]+)?(domain)?$"`.

`yshift` Shifts the position of the whole annotation and arrow up (positive) or down (negative) by this many pixels. Number.

`row` Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
secondary_y	If TRUE, use the secondary y axis of the subplot.
exclude_empty_subplots	If TRUE, skip subplots that have no traces or layout objects.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
color	Color: a CSS color string or R color name.
family	HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.
lineposition	Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".
shadow	Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.
size	Number in the interval [1, Inf].
style	Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".
textcase	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".
variant	Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
weight	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

Details

go_layout_annotation() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_annotation() creates an object and adds it to a figure.

Nested objects documented here:

go_layout_annotation_font() layout.annotation.font: Sets the annotation text font.

go_layout_annotation_hoverlabel() layout.annotation.hoverlabel

go_layout_annotation_hoverlabel_font() layout.annotation.hoverlabel.font: Sets the hover label text font. By default uses the global hover font and size, with color from 'hoverlabel.bordercolor'.

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

---

add_funnel *Funnel traces*

Description

Visualize stages in a process using length-encoded bars. This trace can be used to show data in either a part-to-whole representation wherein each item appears in a single stage, or in a "drop-off" representation wherein each item appears in each stage it traversed. See also the "funnelarea" trace type for a different approach to visualizing funnel data.

Usage

```
add_funnel(fig, ...)

## Default S3 method:
add_funnel(
  fig,
  ...,
  data = NULL,
  alignmentgroup = NULL,
  cliponaxis = NULL,
  connector = NULL,
  constraintext = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextanchor = NULL,
  insidetextfont = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  offset = NULL,
  offsetgroup = NULL,
```

```
  opacity = NULL,
  orientation = NULL,
  outsidetextfont = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textangle = NULL,
  textfont = NULL,
  textinfo = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  width = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  yperiod = NULL,
  yperiod0 = NULL,
  yperiodalignment = NULL,
  zorder = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_funnel(
  arg = NULL,
  ...,
  alignmentgroup = NULL,
  cliponaxis = NULL,
  connector = NULL,
  constraintext = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  hoverinfo = NULL,
```

```
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextanchor = NULL,
  insidetextfont = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  offset = NULL,
  offsetgroup = NULL,
  opacity = NULL,
  orientation = NULL,
  outsidetextfont = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textangle = NULL,
  textfont = NULL,
  textinfo = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  width = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  yperiod = NULL,
  yperiod0 = NULL,
  yperiodalignment = NULL,
  zorder = NULL,
```

```
  skip_invalid = FALSE
)

go_funnel_connector(
  arg = NULL,
  ...,
  fillcolor = NULL,
  line = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_funnel_connector_line(
  arg = NULL,
  ...,
  color = NULL,
  dash = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_funnel_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_funnel_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

```
  go_funnel_insidetextfont(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_funnel_legendgrouptitle(
    arg = NULL,
    ...,
    font = NULL,
    text = NULL,
    skip_invalid = FALSE
  )

  go_funnel_legendgrouptitle_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_funnel_marker(
    arg = NULL,
    ...,
    autocolorscale = NULL,
    cauto = NULL,
    cmax = NULL,
    cmid = NULL,
    cmin = NULL,
    color = NULL,
    coloraxis = NULL,
```

```
  colorbar = NULL,
  colorscale = NULL,
  line = NULL,
  opacity = NULL,
  reversescale = NULL,
  showscale = NULL,
  skip_invalid = FALSE
)

go_funnel_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
```

```
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_funnel_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_funnel_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_funnel_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)
```

```
  go_funnel_marker_colorbar_title_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_funnel_marker_line(
    arg = NULL,
    ...,
    autocolorscale = NULL,
    cauto = NULL,
    cmax = NULL,
    cmid = NULL,
    cmin = NULL,
    color = NULL,
    coloraxis = NULL,
    colorscale = NULL,
    reversescale = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_funnel_outsidetextfont(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_funnel_textfont(
    arg = NULL,
```

```
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
alignmentgroup	Set several traces linked to the same position axis or matching axes to the same alignmentgroup. This controls whether bars compute their positional range dependently or independently. String.
cliponaxis	Determines whether the text nodes are clipped about the subplot axes. To show the text nodes above axis lines and tick labels, make sure to set ‘xaxis.layer‘ and ‘yaxis.layer‘ to "below traces". Logical.
connector	Named list or go_funnel_connector() object.
constraintext	Constrain the size of text inside or outside a bar to be no larger than the bar itself. One of "inside", "outside", "both", "none".
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
dx	Sets the x coordinate step. See ‘x0‘ for more info. Number.
dy	Sets the y coordinate step. See ‘y0‘ for more info. Number.
hoverinfo	Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of "name", "x", "y", "text", "percent initial", "percent previous", "percent total" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_funnel_hoverlabel() object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variables ‘percentInitial‘, ‘percentPrevious‘ and ‘percentTotal‘. Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example ‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

`hovertemplatefallback`
Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Sets hover text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. To be seen, trace ‘hoverinfo‘ must contain a "text" flag. String, or a vector of these.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`insidetextanchor`
Determines if texts are kept at center or start/end points in ‘textposition‘ "inside" mode. One of `"end"`, `"middle"`, `"start"`.

`insidetextfont` Sets the font used for ‘text‘ lying inside the bar. Named list or `go_funnel_insidetextfont()` object.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
Named list or `go_funnel_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`marker` Named list or `go_funnel_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_funnel()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_funnel_marker_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`offset` Shifts the position where the bar is drawn (in position axis units). In "group" barmode, traces that set "offset" will be excluded and drawn in "overlay" mode instead. Number.

`offsetgroup` Set several traces linked to the same position axis or matching axes to the same offsetgroup where bars of the same position coordinate will line up. String.

`opacity` `go_funnel()`: Sets the opacity of the trace. Number in the interval [0, 1].

`go_funnel_marker()`: Sets the opacity of the bars. Number in the interval [0, 1], or a vector of these.

`orientation` `go_funnel()`: Sets the orientation of the funnels. With "v" ("h"), the value of the each bar spans along the vertical (horizontal). By default funnels are tend to be oriented horizontally; unless only "y" array is presented or orientation is set to "v". Also regarding graphs including only ’horizontal’ funnels, "autorange" on the "y-axis" are set to "reversed". One of `"v"`, `"h"`.

`go_funnel_marker_colorbar()`: Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outsidetextfont`

Sets the font used for ‘text‘ lying outside the bar. Named list or `go_funnel_outsidetextfont()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection

where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`text` `go_funnel()`: Sets text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_funnel_legendgrouptitle()`: Sets the title of the legend group. String.

`go_funnel_marker_colorbar_title()`: Sets the title of the color bar. String.

`textangle` Sets the angle of the tick labels with respect to the bar. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. With "auto" the texts may automatically be rotated to fit with the maximum size in bars. Angle in degrees.

`textfont` Sets the font used for ‘text‘. Named list or `go_funnel_textfont()` object.

`textinfo` Determines what trace information appears on the graph. In the case of having multiple funnels, percentages & totals are computed separately (per trace). Flags are rendered in a fixed order; use ‘texttemplate‘ if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of `"label"`, `"text"`, `"percent initial"`, `"percent previous"`, `"percent total"`, `"value"` joined with `"+"`, or one of `"none"`.

`textposition` Specifies the location of the ‘text‘. "inside" positions ‘text‘ inside, next to the bar end (rotated and scaled if needed). "outside" positions ‘text‘ outside, next to the bar end (scaled if needed), unless there is another bar stacked on this one, then the text gets pushed inside. "auto" tries to position ‘text‘ inside the bar, but if the bar is too small and no bar is stacked on this one the text is moved outside. If "none", no text appears. One of `"inside"`, `"outside"`, `"auto"`, `"none"`, or a vector of these.

`texttemplate` Template string used for rendering the information text that appears on points. Note that this will override ‘textinfo‘. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variables ‘percentInitial‘, ‘percentPrevious‘, ‘percentTotal‘, ‘label‘ and ‘value‘. String, or a vector of these.

`texttemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`visible` `go_funnel()`: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`go_funnel_connector()`: Determines if connector regions and lines are drawn. Logical.

`width` `go_funnel()`: Sets the bar width (in position axis units). Number in the interval [0, Inf].

`go_funnel_connector_line()`: Sets the line width (in px). Number in the interval [0, Inf].

`go_funnel_marker_line()`: Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

`x` `go_funnel()`: Sets the x coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_funnel_marker_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`x0` Alternate to ‘x‘. Builds a linear space of x coordinates. Use with ‘dx‘ where ‘x0‘ is the starting coordinate and ‘dx‘ the step. Any value.

`xaxis` Sets a reference between this trace’s x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

`xhoverformat` Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

xperiod — Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the x axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

xperiod0 — Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the x0 axis. When ‘x0period‘ is round number of weeks, the ‘x0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

xperiodalignment — Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the x axis. One of `"start"`, `"middle"`, `"end"`.

y — `go_funnel()`: Sets the y coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_funnel_marker_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

y0 — Alternate to ‘y‘. Builds a linear space of y coordinates. Use with ‘dy‘ where ‘y0‘ is the starting coordinate and ‘dy‘ the step. Any value.

yaxis — Sets a reference between this trace’s y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...

yhoverformat — Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

yperiod — Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the y axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

yperiod0 — Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the y0 axis. When ‘y0period‘ is round number of weeks, the ‘y0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

yperiodalignment — Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the y axis. One of `"start"`, `"middle"`, `"end"`.

zorder — Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher ‘zorder‘ appear in front of those with lower ‘zorder‘. Integer.

row	Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
secondary_y	If TRUE, use the secondary y axis of the subplot.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
fillcolor	Sets the fill color. Color: a CSS color string or R color name.
line	go_funnel_connector(): Named list or go_funnel_connector_line() object. go_funnel_marker(): Named list or go_funnel_marker_line() object.
color	go_funnel_connector_line(): Sets the line color. Color: a CSS color string or R color name. go_funnel_hoverlabel_font(), go_funnel_insidetextfont(), go_funnel_outsidetextfont(), go_funnel_textfont(): Color: a CSS color string or R color name, or a vector of these. go_funnel_legendgrouptitle_font(), go_funnel_marker_colorbar_tickfont(), go_funnel_marker_colorbar_title_font(): Color: a CSS color string or R color name. go_funnel_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.cmin' and 'marker.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by funnel.marker.colorscale, or a vector of these. go_funnel_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.line.cmin' and 'marker.line.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by funnel.marker.line.colorscale, or a vector of these.
dash	Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$".
align	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.
bgcolor	go_funnel_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these. go_funnel_marker_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name.
bordercolor	go_funnel_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

go_funnel_marker_colorbar(): Sets the axis line color. Color: a CSS color string or R color name.

font

go_funnel_hoverlabel(): Sets the font used in hover labels. Named list or go_funnel_hoverlabel_font() object.

go_funnel_legendgrouptitle(): Sets this legend group's title font. Named list or go_funnel_legendgrouptitle_font() object.

go_funnel_marker_colorbar_title(): Sets this color bar's title font. Named list or go_funnel_marker_colorbar_title_font() object.

namelength

Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow

Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

family

go_funnel_hoverlabel_font(), go_funnel_insidetextfont(), go_funnel_outsidetextfont(), go_funnel_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

go_funnel_legendgrouptitle_font(), go_funnel_marker_colorbar_tickfont(), go_funnel_marker_colorbar_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition

go_funnel_hoverlabel_font(), go_funnel_insidetextfont(), go_funnel_outsidetextfont(), go_funnel_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_funnel_legendgrouptitle_font(), go_funnel_marker_colorbar_tickfont(), go_funnel_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow

go_funnel_hoverlabel_font(), go_funnel_insidetextfont(), go_funnel_outsidetextfont(), go_funnel_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_funnel_legendgrouptitle_font(), go_funnel_marker_colorbar_tickfont(), go_funnel_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text

font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

`size` `go_funnel_hoverlabel_font()`, `go_funnel_insidetextfont()`, `go_funnel_outsidetextfont()`, `go_funnel_textfont()`: Number in the interval [1, Inf], or a vector of these.

`go_funnel_legendgrouptitle_font()`, `go_funnel_marker_colorbar_tickfont()`, `go_funnel_marker_colorbar_title_font()`: Number in the interval [1, Inf].

`style` `go_funnel_hoverlabel_font()`, `go_funnel_insidetextfont()`, `go_funnel_outsidetextfont()`, `go_funnel_textfont()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`, or a vector of these.

`go_funnel_legendgrouptitle_font()`, `go_funnel_marker_colorbar_tickfont()`, `go_funnel_marker_colorbar_title_font()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

`textcase` `go_funnel_hoverlabel_font()`, `go_funnel_insidetextfont()`, `go_funnel_outsidetextfont()`, `go_funnel_textfont()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`, or a vector of these.

`go_funnel_legendgrouptitle_font()`, `go_funnel_marker_colorbar_tickfont()`, `go_funnel_marker_colorbar_title_font()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.

`variant` `go_funnel_hoverlabel_font()`, `go_funnel_insidetextfont()`, `go_funnel_outsidetextfont()`, `go_funnel_textfont()`: Sets the variant of the font. One of `"normal"`, `"small-caps"`, `"all-small-caps"`, `"all-petite-caps"`, `"petite-caps"`, `"unicase"`, or a vector of these.

`go_funnel_legendgrouptitle_font()`, `go_funnel_marker_colorbar_tickfont()`, `go_funnel_marker_colorbar_title_font()`: Sets the variant of the font. One of `"normal"`, `"small-caps"`, `"all-small-caps"`, `"all-petite-caps"`, `"petite-caps"`, `"unicase"`.

`weight` `go_funnel_hoverlabel_font()`, `go_funnel_insidetextfont()`, `go_funnel_outsidetextfont()`, `go_funnel_textfont()`: Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

`go_funnel_legendgrouptitle_font()`, `go_funnel_marker_colorbar_tickfont()`, `go_funnel_marker_colorbar_title_font()`: Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

`autocolorscale` `go_funnel_marker()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`go_funnel_marker_line()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`cauto` `go_funnel_marker()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

`go_funnel_marker_line()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

`cmax` `go_funnel_marker()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

`go_funnel_marker_line()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

`cmid` `go_funnel_marker()`: Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

`go_funnel_marker_line()`: Sets the mid-point of the color domain by scaling ‘marker.line.cmin‘ and/or ‘marker.line.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

`cmin` `go_funnel_marker()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`go_funnel_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_funnel_marker_colorbar()` object.

`colorscale` `go_funnel_marker()`: Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list:

Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Re
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_funnel_marker_line()`: Sets the colorscale. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.line.cmin‘ and ‘marker.line.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`reversesescale`

`go_funnel_marker()`: Reverses the color mapping if true. Has an effect only if in ‘marker.color‘ is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.

`go_funnel_marker_line()`: Reverses the color mapping if true. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. If true, ‘marker.line.cmin‘ will correspond to the last color in the array and ‘marker.line.cmax‘ will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘marker.color‘ is set to a numerical array. Logical.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_funnel_marker_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops` List of named lists or `go_funnel_marker_colorbar_tickformatstop()` objects.

`ticklabeloverflow` Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition` Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_funnel_marker_colorbar_title()` object.

`xanchor` Sets this color bar's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the color bar. Defaults to "left" when 'orientation' is "v" and "center" when 'orientation' is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container 'x' refers to. "container" spans the entire 'width' of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`yanchor` Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of `"top"`, `"middle"`, `"bottom"`.

`ypad` Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

`yref` Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

`dtickrange` range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

`enabled` Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

`templateitemname` Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

`value` string - dtickformat for described zoom level, the same as "tickformat" String.

`side` Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of `"right"`, `"top"`, `"bottom"`.

Details

`go_funnel()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_funnel()` creates an object and adds it to a figure.

Nested objects documented here:

`go_funnel_connector()` `funnel.connector`

`go_funnel_connector_line()` `funnel.connector.line`

`go_funnel_hoverlabel()` `funnel.hoverlabel`

`go_funnel_hoverlabel_font()` `funnel.hoverlabel.font`: Sets the font used in hover labels.

`go_funnel_insidetextfont()` `funnel.insidetextfont`: Sets the font used for ‘text‘ lying inside the bar.

`go_funnel_legendgrouptitle()` `funnel.legendgrouptitle`

`go_funnel_legendgrouptitle_font()` `funnel.legendgrouptitle.font`: Sets this legend group’s title font.

`go_funnel_marker()` `funnel.marker`

`go_funnel_marker_colorbar()` `funnel.marker.colorbar`

`go_funnel_marker_colorbar_tickfont()` `funnel.marker.colorbar.tickfont`: Sets the color bar’s tick label font

`go_funnel_marker_colorbar_tickformatstop()` `funnel.marker.colorbar.tickformatstop`

`go_funnel_marker_colorbar_title()` `funnel.marker.colorbar.title`

`go_funnel_marker_colorbar_title_font()` `funnel.marker.colorbar.title.font`: Sets this color bar’s title font.

`go_funnel_marker_line()` `funnel.marker.line`

`go_funnel_outsidetextfont()` `funnel.outsidetextfont`: Sets the font used for ‘text‘ lying outside the bar.

`go_funnel_textfont()` `funnel.textfont`: Sets the font used for ‘text‘.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_funnel(x = 1:3, y = c(2, 1, 3), marker_color = "red")
```

add_funnelarea *Funnelarea traces*

Description

Visualize stages in a process using area-encoded trapezoids. This trace can be used to show data in a part-to-whole representation similar to a "pie" trace, wherein each item appears in a single stage. See also the "funnel" trace type for a different approach to visualizing funnel data.

Usage

```
add_funnelarea(fig, ...)

## Default S3 method:
add_funnelarea(
  fig,
  ...,
  data = NULL,
  aspectratio = NULL,
  baseratio = NULL,
  customdata = NULL,
  dlabel = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextfont = NULL,
  label0 = NULL,
  labels = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  scalegroup = NULL,
  showlegend = NULL,
  text = NULL,
  textfont = NULL,
  textinfo = NULL,
  textposition = NULL,
```

```
  texttemplate = NULL,
  texttemplatefallback = NULL,
  title = NULL,
  uid = NULL,
  uirevision = NULL,
  values = NULL,
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_funnelarea(
  arg = NULL,
  ...,
  aspectratio = NULL,
  baseratio = NULL,
  customdata = NULL,
  dlabel = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextfont = NULL,
  label0 = NULL,
  labels = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  scalegroup = NULL,
  showlegend = NULL,
  text = NULL,
  textfont = NULL,
  textinfo = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  title = NULL,
  uid = NULL,
```

```
  uirevision = NULL,
  values = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_funnelarea_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_funnelarea_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_funnelarea_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_funnelarea_insidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
```

```
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_funnelarea_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_funnelarea_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_funnelarea_marker(
  arg = NULL,
  ...,
  colors = NULL,
  line = NULL,
  pattern = NULL,
  skip_invalid = FALSE
)

go_funnelarea_marker_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
```

```
)

go_funnelarea_marker_pattern(
  arg = NULL,
  ...,
  bgcolor = NULL,
  fgcolor = NULL,
  fgopacity = NULL,
  fillmode = NULL,
  path = NULL,
  shape = NULL,
  size = NULL,
  solidity = NULL,
  skip_invalid = FALSE
)

go_funnelarea_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_funnelarea_title(
  arg = NULL,
  ...,
  font = NULL,
  position = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_funnelarea_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
```

```
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
aspectratio	Sets the ratio between height and width Number in the interval [0, Inf].
baseratio	Sets the ratio between bottom length and maximum top length. Number in the interval [0, 1].
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
dlabel	Sets the label step. See 'label0' for more info. Number.
domain	Named list or go_funnelarea_domain() object.
hoverinfo	Determines what trace information appears on hover. Flags are rendered in a fixed order; use 'hovertemplate' if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of "label", "text", "value", "percent", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_funnelarea_hoverlabel() object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables

with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variables ‘label‘, ‘color‘, ‘value‘, ‘text‘ and ‘percent‘. Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example ‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each sector. If a single string, the same string appears for all data points. If an array of string, the items are mapped in order of this trace's sectors. To be seen, trace ‘hoverinfo‘ must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`insidetextfont`
: Sets the font used for ‘textinfo‘ lying inside the sector. Named list or `go_funnelarea_insidetextfont()` object.

`label0`
: Alternate to ‘labels‘. Builds a numeric set of labels. Use with ‘dlabel‘ where ‘label0‘ is the starting label and ‘dlabel‘ the step. Number.

`labels`
: Sets the sector labels. If ‘labels‘ entries are duplicated, we sum associated ‘values‘ or simply count occurrences if ‘values‘ is not provided. For other array attributes (including color) we use the first non-empty entry among all occurrences of the label. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_funnelarea_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`marker` Named list or `go_funnelarea_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` Sets the trace name. The trace name appears as the legend item and on hover. String.

`opacity` Sets the opacity of the trace. Number in the interval [0, 1].

`scalegroup` If there are multiple funnelareas that should be sized according to their totals, link them by providing a non-empty group id here shared by every trace in the same group. String.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`text` `go_funnelarea()`: Sets text elements associated with each sector. If trace ‘textinfo‘ contains a "text" flag, these elements will be seen on the chart. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_funnelarea_legendgrouptitle()`: Sets the title of the legend group. String.

`go_funnelarea_title()`: Sets the title of the chart. If it is empty, no title is displayed. String.

`textfont` Sets the font used for ‘textinfo‘. Named list or `go_funnelarea_textfont()` object.

`textinfo` Determines what trace information appears on the graph. Flags are rendered in a fixed order; use ‘texttemplate‘ if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of `"label"`, `"text"`, `"value"`, `"percent"` joined with `"+"`, or one of `"none"`.

`textposition` Specifies the location of the ‘textinfo‘. One of `"inside"`, `"none"`, or a vector of these.

`texttemplate` Template string used for rendering the information text that appears on points. Note that this will override ‘textinfo‘. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally,

the template string has access to variables ‘label‘, ‘color‘, ‘value‘, ‘text‘ and ‘percent‘. String, or a vector of these.

`texttemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`title`
: String (the title text), named list or `go_funnelarea_title()` object.

`uid`
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision`
: Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`values`
: Sets the values of the sectors. If omitted, we count occurrences of each label. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`visible`
: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`row`
: `add_funnelarea()`: Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

  `go_funnelarea_domain()`: If there is a layout grid, use the domain for this row in the grid for this funnelarea trace. Integer in the interval [0, Inf].

`col`
: Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`skip_invalid`
: If `TRUE`, silently drop invalid properties instead of raising an error.

`arg`
: Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

`column`
: If there is a layout grid, use the domain for this column in the grid for this funnelarea trace. Integer in the interval [0, Inf].

`x`
: Sets the horizontal domain of this funnelarea trace (in plot fraction). Vector or list of 2 element(s).

`y`
: Sets the vertical domain of this funnelarea trace (in plot fraction). Vector or list of 2 element(s).

`align`
: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

bgcolor
: go_funnelarea_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.
go_funnelarea_marker_pattern(): When there is no colorscale sets the color of background pattern fill. Defaults to a 'marker.color' background when 'fillmode' is "overlay". Otherwise, defaults to a transparent background. Color: a CSS color string or R color name, or a vector of these.

bordercolor
: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

font
: go_funnelarea_hoverlabel(): Sets the font used in hover labels. Named list or go_funnelarea_hoverlabel_font() object.
go_funnelarea_legendgrouptitle(): Sets this legend group's title font. Named list or go_funnelarea_legendgrouptitle_font() object.
go_funnelarea_title(): Sets the font used for 'title'. Named list or go_funnelarea_title_font() object.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color
: go_funnelarea_hoverlabel_font(),go_funnelarea_insidetextfont(),go_funnelarea_textfon
go_funnelarea_title_font(): Color: a CSS color string or R color name, or a vector of these.
go_funnelarea_legendgrouptitle_font(): Color: a CSS color string or R color name.
go_funnelarea_marker_line(): Sets the color of the line enclosing each sector. Defaults to the 'paper_bgcolor' value. Color: a CSS color string or R color name, or a vector of these.

family
: go_funnelarea_hoverlabel_font(),go_funnelarea_insidetextfont(),go_funnelarea_textfon
go_funnelarea_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.
go_funnelarea_legendgrouptitle_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: go_funnelarea_hoverlabel_font(),go_funnelarea_insidetextfont(),go_funnelarea_textfon
go_funnelarea_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_funnelarea_legendgrouptitle_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow

go_funnelarea_hoverlabel_font(), go_funnelarea_insidetextfont(), go_funnelarea_textfon
go_funnelarea_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_funnelarea_legendgrouptitle_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size

go_funnelarea_hoverlabel_font(), go_funnelarea_insidetextfont(), go_funnelarea_textfon
go_funnelarea_title_font(): Number in the interval [1, Inf], or a vector of these.

go_funnelarea_legendgrouptitle_font(): Number in the interval [1, Inf].

go_funnelarea_marker_pattern(): Sets the size of unit squares of the pattern fill in pixels, which corresponds to the interval of repetition of the pattern. Number in the interval [0, Inf], or a vector of these.

style

go_funnelarea_hoverlabel_font(), go_funnelarea_insidetextfont(), go_funnelarea_textfon
go_funnelarea_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_funnelarea_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase

go_funnelarea_hoverlabel_font(), go_funnelarea_insidetextfont(), go_funnelarea_textfon
go_funnelarea_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_funnelarea_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant

go_funnelarea_hoverlabel_font(), go_funnelarea_insidetextfont(), go_funnelarea_textfon
go_funnelarea_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_funnelarea_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight

go_funnelarea_hoverlabel_font(), go_funnelarea_insidetextfont(), go_funnelarea_textfon
go_funnelarea_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_funnelarea_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

colors	Sets the color of each sector. If not specified, the default trace color set is used to pick the sector colors. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
line	Named list or go_funnelarea_marker_line() object.
pattern	Sets the pattern within the marker. Named list or go_funnelarea_marker_pattern() object.
width	Sets the width (in px) of the line enclosing each sector. Number in the interval [0, Inf], or a vector of these.
fgcolor	When there is no colorscale sets the color of foreground pattern fill. Defaults to a 'marker.color' background when 'fillmode' is "replace". Otherwise, defaults to dark grey or white to increase contrast with the 'bgcolor'. Color: a CSS color string or R color name, or a vector of these.
fgopacity	Sets the opacity of the foreground pattern fill. Defaults to a 0.5 when 'fillmode' is "overlay". Otherwise, defaults to 1. Number in the interval [0, 1].
fillmode	Determines whether 'marker.color' should be used as a default to 'bgcolor' or a 'fgcolor'. One of "replace", "overlay".
path	Sets a custom path for pattern fill. Use with no 'shape' or 'solidity', provide an SVG path string for the regions of the square from (0,0) to ('size','size') to color. String, or a vector of these.
shape	Sets the shape of the pattern fill. By default, no pattern is used for filling the area. One of "", "/", "\\", "x", "-", "|", "+", ".", or a vector of these.
solidity	Sets the solidity of the pattern fill. Solidity is roughly the fraction of the area filled by the pattern. Solidity of 0 shows only the background color without pattern and solidty of 1 shows only the foreground color without pattern. Number in the interval [0, 1], or a vector of these.
position	Specifies the location of the 'title'. One of "top left", "top center", "top right".

Details

go_funnelarea() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_funnelarea() creates an object and adds it to a figure.

Nested objects documented here:

go_funnelarea_domain() funnelarea.domain

go_funnelarea_hoverlabel() funnelarea.hoverlabel

go_funnelarea_hoverlabel_font() funnelarea.hoverlabel.font: Sets the font used in hover labels.

go_funnelarea_insidetextfont() funnelarea.insidetextfont: Sets the font used for 'textinfo' lying inside the sector.

go_funnelarea_legendgrouptitle() funnelarea.legendgrouptitle

go_funnelarea_legendgrouptitle_font() funnelarea.legendgrouptitle.font: Sets this legend group's title font.

`go_funnelarea_marker()` `funnelarea.marker`

`go_funnelarea_marker_line()` `funnelarea.marker.line`

`go_funnelarea_marker_pattern()` `funnelarea.marker.pattern`: Sets the pattern within the marker.

`go_funnelarea_textfont()` `funnelarea.textfont`: Sets the font used for 'textinfo'.

`go_funnelarea_title()` `funnelarea.title`

`go_funnelarea_title_font()` `funnelarea.title.font`: Sets the font used for 'title'.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_funnelarea()
```

`add_heatmap` *Heatmap traces*

Description

The data that describes the heatmap value-to-color mapping is set in 'z'. Data in 'z' can either be a matrix of values (ragged or not) or a 1D array of values. In the case where 'z' is a matrix, say that 'z' has N rows and M columns. Then, by default, the resulting heatmap will have N partitions along the y axis and M partitions along the x axis. In other words, the i-th row/ j-th column cell in 'z' is mapped to the i-th partition of the y axis (starting from the bottom of the plot) and the j-th partition of the x-axis (starting from the left of the plot). This behavior can be flipped by using 'transpose'. Moreover, 'x' ('y') can be provided with M or M+1 (N or N+1) elements. If M (N), then the coordinates correspond to the center of the heatmap cells and the cells have equal width. If M+1 (N+1), then the coordinates correspond to the edges of the heatmap cells. In the case where 'z' is a 1D vector, the x and y coordinates must be provided in 'x' and 'y' respectively to form data triplets.

Usage

```
add_heatmap(fig, ...)

## Default S3 method:
add_heatmap(
  fig,
  ...,
```

```
    data = NULL,
    autocolorscale = NULL,
    coloraxis = NULL,
    colorbar = NULL,
    colorscale = NULL,
    connectgaps = NULL,
    customdata = NULL,
    dx = NULL,
    dy = NULL,
    hoverinfo = NULL,
    hoverlabel = NULL,
    hoverongaps = NULL,
    hovertemplate = NULL,
    hovertemplatefallback = NULL,
    hovertext = NULL,
    ids = NULL,
    legend = NULL,
    legendgroup = NULL,
    legendgrouptitle = NULL,
    legendrank = NULL,
    legendwidth = NULL,
    meta = NULL,
    name = NULL,
    opacity = NULL,
    reversescale = NULL,
    showlegend = NULL,
    showscale = NULL,
    text = NULL,
    textfont = NULL,
    texttemplate = NULL,
    texttemplatefallback = NULL,
    transpose = NULL,
    uid = NULL,
    uirevision = NULL,
    visible = NULL,
    x = NULL,
    x0 = NULL,
    xaxis = NULL,
    xcalendar = NULL,
    xgap = NULL,
    xhoverformat = NULL,
    xperiod = NULL,
    xperiod0 = NULL,
    xperiodalignment = NULL,
    xtype = NULL,
    y = NULL,
    y0 = NULL,
    yaxis = NULL,
```

```
    ycalendar = NULL,
    ygap = NULL,
    yhoverformat = NULL,
    yperiod = NULL,
    yperiod0 = NULL,
    yperiodalignment = NULL,
    ytype = NULL,
    z = NULL,
    zauto = NULL,
    zhoverformat = NULL,
    zmax = NULL,
    zmid = NULL,
    zmin = NULL,
    zorder = NULL,
    zsmooth = NULL,
    row = NULL,
    col = NULL,
    secondary_y = NULL,
    skip_invalid = FALSE
  )

  go_heatmap(
    arg = NULL,
    ...,
    autocolorscale = NULL,
    coloraxis = NULL,
    colorbar = NULL,
    colorscale = NULL,
    connectgaps = NULL,
    customdata = NULL,
    dx = NULL,
    dy = NULL,
    hoverinfo = NULL,
    hoverlabel = NULL,
    hoverongaps = NULL,
    hovertemplate = NULL,
    hovertemplatefallback = NULL,
    hovertext = NULL,
    ids = NULL,
    legend = NULL,
    legendgroup = NULL,
    legendgrouptitle = NULL,
    legendrank = NULL,
    legendwidth = NULL,
    meta = NULL,
    name = NULL,
    opacity = NULL,
    reversescale = NULL,
```

```
  showlegend = NULL,
  showscale = NULL,
  text = NULL,
  textfont = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  transpose = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xcalendar = NULL,
  xgap = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  xtype = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  ycalendar = NULL,
  ygap = NULL,
  yhoverformat = NULL,
  yperiod = NULL,
  yperiod0 = NULL,
  yperiodalignment = NULL,
  ytype = NULL,
  z = NULL,
  zauto = NULL,
  zhoverformat = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  zorder = NULL,
  zsmooth = NULL,
  skip_invalid = FALSE
)

go_heatmap_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
```

```
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_heatmap_colorbar_tickfont(
  arg = NULL,
  ...,
```

```
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_heatmap_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_heatmap_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_heatmap_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_heatmap_hoverlabel(
```

```
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_heatmap_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_heatmap_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_heatmap_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

```
go_heatmap_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
autocolorscale	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
coloraxis	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: "coloraxis", "coloraxis2", "coloraxis3", ...
colorbar	Named list or go_heatmap_colorbar() object.
colorscale	Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘zmin‘ and ‘zmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

`connectgaps` Determines whether or not gaps (i.e. {nan} or missing values) in the ‘z‘ data are filled in. It is defaulted to true if ‘z‘ is a one dimensional array and ‘zsmooth‘ is not false; otherwise it is defaulted to false. Logical.

`customdata` Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`dx` Sets the x coordinate step. See ‘x0‘ for more info. Number.

`dy` Sets the y coordinate step. See ‘y0‘ for more info. Number.

`hoverinfo` Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_heatmap_hoverlabel()` object.

`hoverongaps` Determines whether or not gaps (i.e. {nan} or missing values) in the ‘z‘ data have hover labels associated with them. Logical.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example ‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

`hovertemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Same as ‘text‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
Named list or `go_heatmap_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and color-bar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_heatmap()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_heatmap_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` Sets the opacity of the trace. Number in the interval [0, 1].

`reversescale` Reverses the color mapping if true. If true, ‘zmin‘ will correspond to the last color in the array and ‘zmax‘ will correspond to the first color. Logical.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Logical.

`text` `go_heatmap()`: Sets the text elements associated with each z value. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_heatmap_colorbar_title()`: Sets the title of the color bar. String.

`go_heatmap_legendgrouptitle()`: Sets the title of the legend group. String.

`textfont` Sets the text font. Named list or `go_heatmap_textfont()` object.

texttemplate Template string used for rendering the information text that appears on points. Note that this will override ‘textinfo‘. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variables ‘x‘, ‘y‘, ‘z‘ and ‘text‘. String.

texttemplatefallback Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

transpose Transposes the z data. Logical.

uid Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

visible Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.

x go_heatmap(): Sets the x coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_heatmap_colorbar(): Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

x0 Alternate to ‘x‘. Builds a linear space of x coordinates. Use with ‘dx‘ where ‘x0‘ is the starting coordinate and ‘dx‘ the step. Any value.

xaxis Sets a reference between this trace's x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: "x", "x2", "x3", ...

xcalendar Sets the calendar system to use with ‘x‘ date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

xgap Sets the horizontal gap (in pixels) between bricks. Number in the interval [0, Inf].

xhoverformat Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

xperiod Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the x axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

xperiod0 Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the x0 axis. When ‘x0period‘ is round number of weeks, the ‘x0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

xperiodalignment

Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the x axis. One of "start", "middle", "end".

xtype If "array", the heatmap's x coordinates are given by "x" (the default behavior when ‘x‘ is provided). If "scaled", the heatmap's x coordinates are given by "x0" and "dx" (the default behavior when ‘x‘ is not provided). One of "array", "scaled".

y go_heatmap(): Sets the y coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_heatmap_colorbar(): Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

y0 Alternate to ‘y‘. Builds a linear space of y coordinates. Use with ‘dy‘ where ‘y0‘ is the starting coordinate and ‘dy‘ the step. Any value.

yaxis Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: "y", "y2", "y3", ...

ycalendar Sets the calendar system to use with ‘y‘ date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

ygap — Sets the vertical gap (in pixels) between bricks. Number in the interval [0, Inf].

yhoverformat — Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

yperiod — Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the y axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

yperiod0 — Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the y0 axis. When ‘y0period‘ is round number of weeks, the ‘y0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

yperiodalignment — Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the y axis. One of `"start"`, `"middle"`, `"end"`.

ytype — If "array", the heatmap’s y coordinates are given by "y" (the default behavior when ‘y‘ is provided) If "scaled", the heatmap’s y coordinates are given by "y0" and "dy" (the default behavior when ‘y‘ is not provided) One of `"array"`, `"scaled"`.

z — Sets the z data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

zauto — Determines whether or not the color domain is computed with respect to the input data (here in ‘z‘) or the bounds set in ‘zmin‘ and ‘zmax‘ Defaults to ‘false‘ when ‘zmin‘ and ‘zmax‘ are set by the user. Logical.

zhoverformat — Sets the hover text formatting rule for ‘z‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

zmax — Sets the upper bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmin‘ must be set as well. Number.

zmid — Sets the mid-point of the color domain by scaling ‘zmin‘ and/or ‘zmax‘ to be equidistant to this point. Value should have the same units as in ‘z‘. Has no effect when ‘zauto‘ is ‘false‘. Number.

zmin — Sets the lower bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmax‘ must be set as well. Number.

zorder — Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher ‘zorder‘ appear in front of those with lower ‘zorder‘. Integer.

zsmooth — Picks a smoothing algorithm use to smooth ‘z‘ data. One of `"fast"`, `"best"`, `FALSE`.

row	Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
secondary_y	If TRUE, use the secondary y axis of the subplot.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
bgcolor	go_heatmap_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name. go_heatmap_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.
bordercolor	go_heatmap_colorbar(): Sets the axis line color. Color: a CSS color string or R color name. go_heatmap_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.
borderwidth	Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].
dtick	Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.
exponentformat	Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".
labelalias	Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.
len	Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].
lenmode	Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of "fraction", "pixels".
minexponent	Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

nticks — Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

orientation — Sets the orientation of the colorbar. One of "h", "v".

outlinecolor — Sets the axis line color. Color: a CSS color string or R color name.

outlinewidth — Sets the width (in px) of the axis line. Number in the interval [0, Inf].

separatethousands — If "true", even 4-digit integers are separated Logical.

showexponent — If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showticklabels — Determines whether or not the tick labels are drawn. Logical.

showtickprefix — If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".

showticksuffix — Same as ‘showtickprefix‘ but for tick suffixes. One of "all", "first", "last", "none".

thickness — Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode — Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of "fraction", "pixels".

tick0 — Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle — Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

tickcolor — Sets the tick color. Color: a CSS color string or R color name.

tickfont — Sets the color bar’s tick label font Named list or go_heatmap_colorbar_tickfont() object.

tickformat — Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops — List of named lists or go_heatmap_colorbar_tickformatstop() objects.

`ticklabeloverflow`
: Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition`
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep`
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen`
: Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode`
: Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix`
: Sets a tick label prefix. String.

`ticks`
: Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix`
: Sets a tick label suffix. String.

`ticktext`
: Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals`
: Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth`
: Sets the tick width (in px). Number in the interval [0, Inf].

`title`
: String (the title text), named list or `go_heatmap_colorbar_title()` object.

`xanchor`
: Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad`
: Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref`
: Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

yanchor — Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

color — go_heatmap_colorbar_tickfont(), go_heatmap_colorbar_title_font(), go_heatmap_legendgrouptitle_font(), go_heatmap_textfont(): Color: a CSS color string or R color name.

go_heatmap_hoverlabel_font(): Color: a CSS color string or R color name, or a vector of these.

family — go_heatmap_colorbar_tickfont(), go_heatmap_colorbar_title_font(), go_heatmap_legendgrouptitle_font(), go_heatmap_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

go_heatmap_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

lineposition — go_heatmap_colorbar_tickfont(), go_heatmap_colorbar_title_font(), go_heatmap_legendgrouptitle_font(), go_heatmap_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

go_heatmap_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

shadow — go_heatmap_colorbar_tickfont(), go_heatmap_colorbar_title_font(), go_heatmap_legendgrouptitle_font(), go_heatmap_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

go_heatmap_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size — go_heatmap_colorbar_tickfont(), go_heatmap_colorbar_title_font(), go_heatmap_legendgrouptitle_font(), go_heatmap_textfont(): Number in the interval [1, Inf].

go_heatmap_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

style — go_heatmap_colorbar_tickfont(), go_heatmap_colorbar_title_font(), go_heatmap_legendgrouptitle_font(), go_heatmap_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

go_heatmap_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

textcase — go_heatmap_colorbar_tickfont(), go_heatmap_colorbar_title_font(), go_heatmap_legendgrouptitle_font(), go_heatmap_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

go_heatmap_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant — go_heatmap_colorbar_tickfont(), go_heatmap_colorbar_title_font(), go_heatmap_legendgrouptitle_font(), go_heatmap_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

go_heatmap_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight — go_heatmap_colorbar_tickfont(), go_heatmap_colorbar_title_font(), go_heatmap_legendgrouptitle_font(), go_heatmap_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

go_heatmap_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

font — go_heatmap_colorbar_title(): Sets this color bar's title font. Named list or go_heatmap_colorbar_title_font() object.

go_heatmap_hoverlabel(): Sets the font used in hover labels. Named list or go_heatmap_hoverlabel_font() object.

	go_heatmap_legendgrouptitle(): Sets this legend group's title font. Named list or go_heatmap_legendgrouptitle_font() object.
side	Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of "right", "top", "bottom".
align	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.
namelength	Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.
showarrow	Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

Details

go_heatmap() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_heatmap() creates an object and adds it to a figure.

Nested objects documented here:

go_heatmap_colorbar() heatmap.colorbar

go_heatmap_colorbar_tickfont() heatmap.colorbar.tickfont: Sets the color bar's tick label font

go_heatmap_colorbar_tickformatstop() heatmap.colorbar.tickformatstop

go_heatmap_colorbar_title() heatmap.colorbar.title

go_heatmap_colorbar_title_font() heatmap.colorbar.title.font: Sets this color bar's title font.

go_heatmap_hoverlabel() heatmap.hoverlabel

go_heatmap_hoverlabel_font() heatmap.hoverlabel.font: Sets the font used in hover labels.

go_heatmap_legendgrouptitle() heatmap.legendgrouptitle

go_heatmap_legendgrouptitle_font() heatmap.legendgrouptitle.font: Sets this legend group's title font.

go_heatmap_textfont() heatmap.textfont: Sets the text font.

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_heatmap(x = 1:3, y = c(2, 1, 3))
```

---

add_histogram *Histogram traces*

Description

The sample data from which statistics are computed is set in 'x' for vertically spanning histograms and in 'y' for horizontally spanning histograms. Binning options are set 'xbins' and 'ybins' respectively if no aggregation data is provided.

Usage

```
add_histogram(fig, ...)

## Default S3 method:
add_histogram(
  fig,
  ...,
  data = NULL,
  alignmentgroup = NULL,
  autobinx = NULL,
  autobiny = NULL,
  bingroup = NULL,
  cliponaxis = NULL,
  constraintext = NULL,
  cumulative = NULL,
  customdata = NULL,
  error_x = NULL,
  error_y = NULL,
  histfunc = NULL,
  histnorm = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextanchor = NULL,
  insidetextfont = NULL,
  legend = NULL,
  legendgroup = NULL,
```

```
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  nbinsx = NULL,
  nbinsy = NULL,
  offsetgroup = NULL,
  opacity = NULL,
  orientation = NULL,
  outsidetextfont = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textangle = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  x = NULL,
  xaxis = NULL,
  xbins = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  y = NULL,
  yaxis = NULL,
  ybins = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  zorder = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_histogram(
  arg = NULL,
  ...,
  alignmentgroup = NULL,
  autobinx = NULL,
  autobiny = NULL,
```

```
bingroup = NULL,
cliponaxis = NULL,
constraintext = NULL,
cumulative = NULL,
customdata = NULL,
error_x = NULL,
error_y = NULL,
histfunc = NULL,
histnorm = NULL,
hoverinfo = NULL,
hoverlabel = NULL,
hovertemplate = NULL,
hovertemplatefallback = NULL,
hovertext = NULL,
ids = NULL,
insidetextanchor = NULL,
insidetextfont = NULL,
legend = NULL,
legendgroup = NULL,
legendgrouptitle = NULL,
legendrank = NULL,
legendwidth = NULL,
marker = NULL,
meta = NULL,
name = NULL,
nbinsx = NULL,
nbinsy = NULL,
offsetgroup = NULL,
opacity = NULL,
orientation = NULL,
outsidetextfont = NULL,
selected = NULL,
selectedpoints = NULL,
showlegend = NULL,
text = NULL,
textangle = NULL,
textfont = NULL,
textposition = NULL,
texttemplate = NULL,
texttemplatefallback = NULL,
uid = NULL,
uirevision = NULL,
unselected = NULL,
visible = NULL,
x = NULL,
xaxis = NULL,
xbins = NULL,
xcalendar = NULL,
```

```
  xhoverformat = NULL,
  y = NULL,
  yaxis = NULL,
  ybins = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  zorder = NULL,
  skip_invalid = FALSE
)

go_histogram_cumulative(
  arg = NULL,
  ...,
  currentbin = NULL,
  direction = NULL,
  enabled = NULL,
  skip_invalid = FALSE
)

go_histogram_error_x(
  arg = NULL,
  ...,
  array = NULL,
  arrayminus = NULL,
  color = NULL,
  copy_ystyle = NULL,
  symmetric = NULL,
  thickness = NULL,
  traceref = NULL,
  tracerefminus = NULL,
  type = NULL,
  value = NULL,
  valueminus = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_histogram_error_y(
  arg = NULL,
  ...,
  array = NULL,
  arrayminus = NULL,
  color = NULL,
  symmetric = NULL,
  thickness = NULL,
  traceref = NULL,
  tracerefminus = NULL,
```

```
  type = NULL,
  value = NULL,
  valueminus = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_histogram_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_histogram_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram_insidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
```

```
)

go_histogram_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_histogram_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram_marker(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  cornerradius = NULL,
  line = NULL,
  opacity = NULL,
  pattern = NULL,
  reversescale = NULL,
  showscale = NULL,
  skip_invalid = FALSE
)

go_histogram_marker_colorbar(
  arg = NULL,
```

```
...,
bgcolor = NULL,
bordercolor = NULL,
borderwidth = NULL,
dtick = NULL,
exponentformat = NULL,
labelalias = NULL,
len = NULL,
lenmode = NULL,
minexponent = NULL,
nticks = NULL,
orientation = NULL,
outlinecolor = NULL,
outlinewidth = NULL,
separatethousands = NULL,
showexponent = NULL,
showticklabels = NULL,
showtickprefix = NULL,
showticksuffix = NULL,
thickness = NULL,
thicknessmode = NULL,
tick0 = NULL,
tickangle = NULL,
tickcolor = NULL,
tickfont = NULL,
tickformat = NULL,
tickformatstops = NULL,
ticklabeloverflow = NULL,
ticklabelposition = NULL,
ticklabelstep = NULL,
ticklen = NULL,
tickmode = NULL,
tickprefix = NULL,
ticks = NULL,
ticksuffix = NULL,
ticktext = NULL,
tickvals = NULL,
tickwidth = NULL,
title = NULL,
x = NULL,
xanchor = NULL,
xpad = NULL,
xref = NULL,
y = NULL,
yanchor = NULL,
ypad = NULL,
yref = NULL,
skip_invalid = FALSE
```

```
)

go_histogram_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_histogram_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_histogram_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
```

```
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram_marker_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_histogram_marker_pattern(
  arg = NULL,
  ...,
  bgcolor = NULL,
  fgcolor = NULL,
  fgopacity = NULL,
  fillmode = NULL,
  path = NULL,
  shape = NULL,
  size = NULL,
  solidity = NULL,
  skip_invalid = FALSE
)

go_histogram_outsidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

```
  go_histogram_selected(
    arg = NULL,
    ...,
    marker = NULL,
    textfont = NULL,
    skip_invalid = FALSE
  )

  go_histogram_selected_marker(
    arg = NULL,
    ...,
    color = NULL,
    opacity = NULL,
    skip_invalid = FALSE
  )

  go_histogram_selected_textfont(
    arg = NULL,
    ...,
    color = NULL,
    skip_invalid = FALSE
  )

  go_histogram_textfont(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_histogram_unselected(
    arg = NULL,
    ...,
    marker = NULL,
    textfont = NULL,
    skip_invalid = FALSE
  )

  go_histogram_unselected_marker(
```

```
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  skip_invalid = FALSE
)

go_histogram_unselected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

go_histogram_xbins(
  arg = NULL,
  ...,
  end = NULL,
  size = NULL,
  start = NULL,
  skip_invalid = FALSE
)

go_histogram_ybins(
  arg = NULL,
  ...,
  end = NULL,
  size = NULL,
  start = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
alignmentgroup	Set several traces linked to the same position axis or matching axes to the same alignmentgroup. This controls whether bars compute their positional range dependently or independently. String.
autobinx	Obsolete: since v1.42 each bin attribute is auto-determined separately and 'autobinx' is not needed. However, we accept 'autobinx: true' or 'false' and will update 'xbins' accordingly before deleting 'autobinx' from the trace. Logical.

`autobiny` Obsolete: since v1.42 each bin attribute is auto-determined separately and ‘autobiny‘ is not needed. However, we accept ‘autobiny: true‘ or ‘false‘ and will update ‘ybins‘ accordingly before deleting ‘autobiny‘ from the trace. Logical.

`bingroup` Set a group of histogram traces which will have compatible bin settings. Note that traces on the same subplot and with the same "orientation" under ‘barmode‘ "stack", "relative" and "group" are forced into the same bingroup, Using ‘bingroup‘, traces under ‘barmode‘ "overlay" and on different axes (of the same axis type) can have compatible bin settings. Note that histogram and histogram2d* trace can share the same ‘bingroup‘ String.

`cliponaxis` Determines whether the text nodes are clipped about the subplot axes. To show the text nodes above axis lines and tick labels, make sure to set ‘xaxis.layer‘ and ‘yaxis.layer‘ to "below traces". Logical.

`constraintext` Constrain the size of text inside or outside a bar to be no larger than the bar itself. One of `"inside"`, `"outside"`, `"both"`, `"none"`.

`cumulative` Named list or `go_histogram_cumulative()` object.

`customdata` Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`error_x` Named list or `go_histogram_error_x()` object.

`error_y` Named list or `go_histogram_error_y()` object.

`histfunc` Specifies the binning function used for this histogram trace. If "count", the histogram values are computed by counting the number of values lying inside each bin. If "sum", "avg", "min", "max", the histogram values are computed using the sum, the average, the minimum or the maximum of the values lying inside each bin respectively. One of `"count"`, `"sum"`, `"avg"`, `"min"`, `"max"`.

`histnorm` Specifies the type of normalization used for this histogram trace. If *", the span of each bar corresponds to the number of occurrences (i.e. the number of data points lying inside the bins). If "percent" / "probability", the span of each bar corresponds to the percentage / fraction of occurrences with respect to the total number of sample points (here, the sum of all bin HEIGHTS equals 100% / 1). If "density", the span of each bar corresponds to the number of occurrences in a bin divided by the size of the bin interval (here, the sum of all bin AREAS equals the total number of sample points). If "probability density*, the area of each bar corresponds to the probability that an event will fall into the corresponding bin (here, the sum of all bin AREAS equals 1). One of `""`, `"percent"`, `"probability"`, `"density"`, `"probability density"`.

`hoverinfo` Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_histogram_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_},

{%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variable 'binNumber' Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Same as 'text'. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`insidetextanchor`
: Determines if texts are kept at center or start/end points in 'textposition' "inside" mode. One of `"end"`, `"middle"`, `"start"`.

`insidetextfont`
: Sets the font used for 'text' lying inside the bar. Named list or `go_histogram_insidetextfont()` object.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_histogram_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`marker` `go_histogram()`: Named list or `go_histogram_marker()` object.

`go_histogram_selected()`: Named list or `go_histogram_selected_marker()` object.

`go_histogram_unselected()`: Named list or `go_histogram_unselected_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_histogram()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_histogram_marker_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`nbinsx` Specifies the maximum number of desired bins. This value will be used in an algorithm that will decide the optimal bin size such that the histogram best visualizes the distribution of the data. Ignored if ‘xbins.size‘ is provided. Integer in the interval [0, Inf].

`nbinsy` Specifies the maximum number of desired bins. This value will be used in an algorithm that will decide the optimal bin size such that the histogram best visualizes the distribution of the data. Ignored if ‘ybins.size‘ is provided. Integer in the interval [0, Inf].

`offsetgroup` Set several traces linked to the same position axis or matching axes to the same offsetgroup where bars of the same position coordinate will line up. String.

`opacity` `go_histogram()`: Sets the opacity of the trace. Number in the interval [0, 1].

`go_histogram_marker()`: Sets the opacity of the bars. Number in the interval [0, 1], or a vector of these.

`go_histogram_selected_marker()`: Sets the marker opacity of selected points. Number in the interval [0, 1].

`go_histogram_unselected_marker()`: Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

`orientation` `go_histogram()`: Sets the orientation of the bars. With "v" ("h"), the value of the each bar spans along the vertical (horizontal). One of `"v"`, `"h"`.

`go_histogram_marker_colorbar()`: Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outsidetextfont`
: Sets the font used for 'text' lying outside the bar. Named list or `go_histogram_outsidetextfont()` object.

`selected`
: Named list or `go_histogram_selected()` object.

`selectedpoints`
: Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the 'unselected' are turned on for all points, whereas, any other non-array values means no selection all where the 'selected' and 'unselected' styles have no effect. Any value.

`showlegend`
: Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`text`
: `go_histogram()`: Sets hover text elements associated with each bar. If a single string, the same string appears over all bars. If an array of string, the items are mapped in order to the this trace's coordinates. String, or a vector of these.

  `go_histogram_legendgrouptitle()`: Sets the title of the legend group. String.

  `go_histogram_marker_colorbar_title()`: Sets the title of the color bar. String.

`textangle`
: Sets the angle of the tick labels with respect to the bar. For example, a 'tickangle' of -90 draws the tick labels vertically. With "auto" the texts may automatically be rotated to fit with the maximum size in bars. Angle in degrees.

`textfont`
: `go_histogram()`: Sets the text font. Named list or `go_histogram_textfont()` object.

  `go_histogram_selected()`: Named list or `go_histogram_selected_textfont()` object.

  `go_histogram_unselected()`: Named list or `go_histogram_unselected_textfont()` object.

`textposition`
: Specifies the location of the 'text'. "inside" positions 'text' inside, next to the bar end (rotated and scaled if needed). "outside" positions 'text' outside, next to the bar end (scaled if needed), unless there is another bar stacked on this one, then the text gets pushed inside. "auto" tries to position 'text' inside the bar, but if the bar is too small and no bar is stacked on this one the text is moved outside. If "none", no text appears. One of `"inside"`, `"outside"`, `"auto"`, `"none"`.

`texttemplate`
: Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'label' and 'value'. String.

texttemplatefallback
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

uid
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision
: Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

unselected
: Named list or `go_histogram_unselected()` object.

visible
: `go_histogram()`: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

  `go_histogram_error_x()`, `go_histogram_error_y()`: Determines whether or not this set of error bars is visible. Logical.

x
: `go_histogram()`: Sets the sample data to be binned on the x axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  `go_histogram_marker_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

xaxis
: Sets a reference between this trace's x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

xbins
: Named list or `go_histogram_xbins()` object.

xcalendar
: Sets the calendar system to use with ‘x‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

xhoverformat
: Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

y
: go_histogram(): Sets the sample data to be binned on the y axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  go_histogram_marker_colorbar(): Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

yaxis
: Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: "y", "y2", "y3", ...

ybins
: Named list or go_histogram_ybins() object.

ycalendar
: Sets the calendar system to use with ‘y‘ date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

yhoverformat
: Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

zorder
: Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher ‘zorder‘ appear in front of those with lower ‘zorder‘. Integer.

row
: Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.

col
: Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

secondary_y
: If TRUE, use the secondary y axis of the subplot.

skip_invalid
: If TRUE, silently drop invalid properties instead of raising an error.

arg
: Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

currentbin
: Only applies if cumulative is enabled. Sets whether the current bin is included, excluded, or has half of its value included in the current cumulative value. "include" is the default for compatibility with various other tools, however it introduces a half-bin bias to the results. "exclude" makes the opposite half-bin bias, and "half" removes it. One of "include", "exclude", "half".

direction
: Only applies if cumulative is enabled. If "increasing" (default) we sum all prior bins, so the result increases from left to right. If "decreasing" we sum later bins so the result decreases from left to right. One of "increasing", "decreasing".

enabled

go_histogram_cumulative(): If true, display the cumulative distribution by summing the binned values. Use the ‘direction‘ and ‘centralbin‘ attributes to tune the accumulation method. Note: in this mode, the "density" ‘histnorm‘ settings behave the same as their equivalents without "density": *" and "density" both rise to the number of data points, and "probability" and "probability density* both rise to the number of sample points. Logical.

go_histogram_marker_colorbar_tickformatstop(): Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

array

Sets the data corresponding the length of each error bar. Values are plotted relative to the underlying data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

arrayminus

Sets the data corresponding the length of each error bar in the bottom (left) direction for vertical (horizontal) bars Values are plotted relative to the underlying data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

color

go_histogram_error_x(), go_histogram_error_y(): Sets the stroke color of the error bars. Color: a CSS color string or R color name.

go_histogram_hoverlabel_font(): Color: a CSS color string or R color name, or a vector of these.

go_histogram_insidetextfont(), go_histogram_legendgrouptitle_font(), go_histogram_marker_colorbar_tickfont(), go_histogram_marker_colorbar_title_font(), go_histogram_outsidetextfont(), go_histogram_textfont(): Color: a CSS color string or R color name.

go_histogram_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.cmin‘ and ‘marker.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by histogram.marker.colorscale, or a vector of these.

go_histogram_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by histogram.marker.line.colorscale, or a vector of these.

go_histogram_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name.

go_histogram_selected_textfont(): Sets the text font color of selected points. Color: a CSS color string or R color name.

go_histogram_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

go_histogram_unselected_textfont(): Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

copy_ystyle

Logical.

`symmetric` Determines whether or not the error bars have the same length in both direction (top/bottom for vertical bars, left/right for horizontal bars. Logical.

`thickness` `go_histogram_error_x()`, `go_histogram_error_y()`: Sets the thickness (in px) of the error bars. Number in the interval [0, Inf].

`go_histogram_marker_colorbar()`: Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`traceref` Integer in the interval [0, Inf].

`tracerefminus` Integer in the interval [0, Inf].

`type` Determines the rule used to generate the error bars. If "constant", the bar lengths are of a constant value. Set this constant in ‘value‘. If "percent", the bar lengths correspond to a percentage of underlying data. Set this percentage in ‘value‘. If "sqrt", the bar lengths correspond to the square of the underlying data. If "data", the bar lengths are set with data set ‘array‘. One of `"percent"`, `"constant"`, `"sqrt"`, `"data"`.

`value` `go_histogram_error_x()`, `go_histogram_error_y()`: Sets the value of either the percentage (if ‘type‘ is set to "percent") or the constant (if ‘type‘ is set to "constant") corresponding to the lengths of the error bars. Number in the interval [0, Inf].

`go_histogram_marker_colorbar_tickformatstop()`: string - dtickformat for described zoom level, the same as "tickformat" String.

`valueminus` Sets the value of either the percentage (if ‘type‘ is set to "percent") or the constant (if ‘type‘ is set to "constant") corresponding to the lengths of the error bars in the bottom (left) direction for vertical (horizontal) bars Number in the interval [0, Inf].

`width` `go_histogram_error_x()`, `go_histogram_error_y()`: Sets the width (in px) of the cross-bar at both ends of the error bars. Number in the interval [0, Inf].

`go_histogram_marker_line()`: Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

`align` Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`bgcolor` `go_histogram_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`go_histogram_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

`go_histogram_marker_pattern()`: When there is no colorscale sets the color of background pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "overlay". Otherwise, defaults to a transparent background. Color: a CSS color string or R color name, or a vector of these.

`bordercolor` `go_histogram_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`go_histogram_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

font
: go_histogram_hoverlabel(): Sets the font used in hover labels. Named list or go_histogram_hoverlabel_font() object.

  go_histogram_legendgrouptitle(): Sets this legend group's title font. Named list or go_histogram_legendgrouptitle_font() object.

  go_histogram_marker_colorbar_title(): Sets this color bar's title font. Named list or go_histogram_marker_colorbar_title_font() object.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

family
: go_histogram_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

  go_histogram_insidetextfont(), go_histogram_legendgrouptitle_font(), go_histogram_marker_colorbar_tickfont(), go_histogram_marker_colorbar_title_font(), go_histogram_outsidetextfont(), go_histogram_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: go_histogram_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

  go_histogram_insidetextfont(), go_histogram_legendgrouptitle_font(), go_histogram_marker_colorbar_tickfont(), go_histogram_marker_colorbar_title_font(), go_histogram_outsidetextfont(), go_histogram_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow
: go_histogram_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

  go_histogram_insidetextfont(), go_histogram_legendgrouptitle_font(), go_histogram_marker_colorbar_tickfont(), go_histogram_marker_colorbar_title_font(), go_histogram_outsidetextfont(), go_histogram_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: go_histogram_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

go_histogram_insidetextfont(), go_histogram_legendgrouptitle_font(), go_histogram_marker_colorbar_tickfont(), go_histogram_marker_colorbar_title_font(), go_histogram_outsidetextfont(), go_histogram_textfont(): Number in the interval [1, Inf].

go_histogram_marker_pattern(): Sets the size of unit squares of the pattern fill in pixels, which corresponds to the interval of repetition of the pattern. Number in the interval [0, Inf], or a vector of these.

go_histogram_xbins(): Sets the size of each x axis bin. Default behavior: If 'nbinsx' is 0 or omitted, we choose a nice round bin size such that the number of bins is about the same as the typical number of samples in each bin. If 'nbinsx' is provided, we choose a nice round bin size giving no more than that many bins. For date data, use milliseconds or "M<n>" for months, as in 'axis.dtick'. For category data, the number of categories to bin together (always defaults to 1). If multiple non-overlaying histograms share a subplot, the first explicit 'size' is used and all others discarded. If no 'size' is provided, the sample data from all traces is combined to determine 'size' as described above. Any value.

go_histogram_ybins(): Sets the size of each y axis bin. Default behavior: If 'nbinsy' is 0 or omitted, we choose a nice round bin size such that the number of bins is about the same as the typical number of samples in each bin. If 'nbinsy' is provided, we choose a nice round bin size giving no more than that many bins. For date data, use milliseconds or "M<n>" for months, as in 'axis.dtick'. For category data, the number of categories to bin together (always defaults to 1). If multiple non-overlaying histograms share a subplot, the first explicit 'size' is used and all others discarded. If no 'size' is provided, the sample data from all traces is combined to determine 'size' as described above. Any value.

style
: go_histogram_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_histogram_insidetextfont(), go_histogram_legendgrouptitle_font(), go_histogram_marker_colorbar_tickfont(), go_histogram_marker_colorbar_title_font(), go_histogram_outsidetextfont(), go_histogram_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_histogram_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_histogram_insidetextfont(), go_histogram_legendgrouptitle_font(), go_histogram_marker_colorbar_tickfont(), go_histogram_marker_colorbar_title_font(), go_histogram_outsidetextfont(), go_histogram_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_histogram_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_histogram_insidetextfont(), go_histogram_legendgrouptitle_font(), go_histogram_marker_colorbar_tickfont(), go_histogram_marker_colorbar_title_font(), go_histogram_outsidetextfont(), go_histogram_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
go_histogram_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_histogram_insidetextfont(), go_histogram_legendgrouptitle_font(), go_histogram_marker_colorbar_tickfont(), go_histogram_marker_colorbar_title_font(), go_histogram_outsidetextfont(), go_histogram_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

autocolorscale
go_histogram_marker(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

go_histogram_marker_line(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

cauto
go_histogram_marker(): Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

go_histogram_marker_line(): Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

cmax
go_histogram_marker(): Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

go_histogram_marker_line(): Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

cmid
go_histogram_marker(): Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

go_histogram_marker_line(): Sets the mid-point of the color domain by scaling ‘marker.line.cmin‘ and/or ‘marker.line.cmax‘ to be equidistant to this

point. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

`cmin` `go_histogram_marker()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`go_histogram_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_histogram_marker_colorbar()` object.

`colorscale` `go_histogram_marker()`: Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Re
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_histogram_marker_line()`: Sets the colorscale. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.line.cmin‘ and ‘marker.line.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,H
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`cornerradius` Sets the rounding of corners. May be an integer number of pixels, or a percentage of bar width (as a string ending in %). Defaults to ‘layout.barcornerradius‘. In stack or relative barmode, the first trace to set cornerradius is used for the whole stack. Any value.

`line` Named list or `go_histogram_marker_line()` object.

`pattern` Sets the pattern within the marker. Named list or `go_histogram_marker_pattern()` object.

`reversescale` `go_histogram_marker()`: Reverses the color mapping if true. Has an effect only if in ‘marker.color‘ is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.

`go_histogram_marker_line()`: Reverses the color mapping if true. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. If true, ‘marker.line.cmin‘ will correspond to the last color in the array and ‘marker.line.cmax‘ will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘marker.color‘ is set to a numerical array. Logical.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

separatethousands
: If "true", even 4-digit integers are separated Logical.

showexponent
: If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticklabels
: Determines whether or not the tick labels are drawn. Logical.

showtickprefix
: If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticksuffix
: Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

thicknessmode
: Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

tick0
: Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle
: Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

tickcolor
: Sets the tick color. Color: a CSS color string or R color name.

tickfont
: Sets the color bar’s tick label font Named list or `go_histogram_marker_colorbar_tickfont()` object.

tickformat
: Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops
: List of named lists or `go_histogram_marker_colorbar_tickformatstop()` objects.

ticklabeloverflow
: Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

ticklabelposition
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is

"v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelstep — Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

ticklen — Sets the tick length (in px). Number in the interval [0, Inf].

tickmode — Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of "auto", "linear", "array".

tickprefix — Sets a tick label prefix. String.

ticks — Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of "outside", "inside", "".

ticksuffix — Sets a tick label suffix. String.

ticktext — Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals — Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth — Sets the tick width (in px). Number in the interval [0, Inf].

title — String (the title text), named list or go_histogram_marker_colorbar_title() object.

xanchor — Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of "left", "center", "right".

xpad — Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".

yanchor — Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

side — Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of "right", "top", "bottom".

fgcolor — When there is no colorscale sets the color of foreground pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "replace". Otherwise, defaults to dark grey or white to increase contrast with the ‘bgcolor‘. Color: a CSS color string or R color name, or a vector of these.

fgopacity — Sets the opacity of the foreground pattern fill. Defaults to a 0.5 when ‘fillmode‘ is "overlay". Otherwise, defaults to 1. Number in the interval [0, 1].

fillmode — Determines whether ‘marker.color‘ should be used as a default to ‘bgcolor‘ or a ‘fgcolor‘. One of "replace", "overlay".

path — Sets a custom path for pattern fill. Use with no ‘shape‘ or ‘solidity‘, provide an SVG path string for the regions of the square from (0,0) to (‘size‘,‘size‘) to color. String, or a vector of these.

shape — Sets the shape of the pattern fill. By default, no pattern is used for filling the area. One of "", "/", "\\", "x", "-", "|", "+", ".", or a vector of these.

solidity — Sets the solidity of the pattern fill. Solidity is roughly the fraction of the area filled by the pattern. Solidity of 0 shows only the background color without pattern and solidty of 1 shows only the foreground color without pattern. Number in the interval [0, 1], or a vector of these.

end — go_histogram_xbins(): Sets the end value for the x axis bins. The last bin may not end exactly at this value, we increment the bin edge by ‘size‘ from ‘start‘ until we reach or exceed ‘end‘. Defaults to the maximum data value. Like ‘start‘, for dates use a date string, and for category data ‘end‘ is based on the category serial numbers. Any value.

go_histogram_ybins(): Sets the end value for the y axis bins. The last bin may not end exactly at this value, we increment the bin edge by ‘size‘ from ‘start‘ until we reach or exceed ‘end‘. Defaults to the maximum data value. Like ‘start‘, for dates use a date string, and for category data ‘end‘ is based on the category serial numbers. Any value.

start — go_histogram_xbins(): Sets the starting value for the x axis bins. Defaults to the minimum data value, shifted down if necessary to make nice round values

and to remove ambiguous bin edges. For example, if most of the data is integers we shift the bin edges 0.5 down, so a 'size' of 5 would have a default 'start' of -0.5, so it is clear that 0-4 are in the first bin, 5-9 in the second, but continuous data gets a start of 0 and bins [0,5), [5,10) etc. Dates behave similarly, and 'start' should be a date string. For category data, 'start' is based on the category serial numbers, and defaults to -0.5. If multiple non-overlaying histograms share a subplot, the first explicit 'start' is used exactly and all others are shifted down (if necessary) to differ from that one by an integer number of bins. Any value.

`go_histogram_ybins()`: Sets the starting value for the y axis bins. Defaults to the minimum data value, shifted down if necessary to make nice round values and to remove ambiguous bin edges. For example, if most of the data is integers we shift the bin edges 0.5 down, so a 'size' of 5 would have a default 'start' of -0.5, so it is clear that 0-4 are in the first bin, 5-9 in the second, but continuous data gets a start of 0 and bins [0,5), [5,10) etc. Dates behave similarly, and 'start' should be a date string. For category data, 'start' is based on the category serial numbers, and defaults to -0.5. If multiple non-overlaying histograms share a subplot, the first explicit 'start' is used exactly and all others are shifted down (if necessary) to differ from that one by an integer number of bins. Any value.

Details

`go_histogram()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_histogram()` creates an object and adds it to a figure.

Nested objects documented here:

`go_histogram_cumulative()` `histogram.cumulative`

`go_histogram_error_x()` `histogram.error_x`

`go_histogram_error_y()` `histogram.error_y`

`go_histogram_hoverlabel()` `histogram.hoverlabel`

`go_histogram_hoverlabel_font()` `histogram.hoverlabel.font`: Sets the font used in hover labels.

`go_histogram_insidetextfont()` `histogram.insidetextfont`: Sets the font used for 'text' lying inside the bar.

`go_histogram_legendgrouptitle()` `histogram.legendgrouptitle`

`go_histogram_legendgrouptitle_font()` `histogram.legendgrouptitle.font`: Sets this legend group's title font.

`go_histogram_marker()` `histogram.marker`

`go_histogram_marker_colorbar()` `histogram.marker.colorbar`

`go_histogram_marker_colorbar_tickfont()` `histogram.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_histogram_marker_colorbar_tickformatstop()` `histogram.marker.colorbar.tickformatstop`

`go_histogram_marker_colorbar_title()` `histogram.marker.colorbar.title`

`go_histogram_marker_colorbar_title_font()` `histogram.marker.colorbar.title.font`: Sets this color bar's title font.

go_histogram_marker_line() histogram.marker.line

go_histogram_marker_pattern() histogram.marker.pattern: Sets the pattern within the marker.

go_histogram_outsidetextfont() histogram.outsidetextfont: Sets the font used for 'text' lying outside the bar.

go_histogram_selected() histogram.selected

go_histogram_selected_marker() histogram.selected.marker

go_histogram_selected_textfont() histogram.selected.textfont

go_histogram_textfont() histogram.textfont: Sets the text font.

go_histogram_unselected() histogram.unselected

go_histogram_unselected_marker() histogram.unselected.marker

go_histogram_unselected_textfont() histogram.unselected.textfont

go_histogram_xbins() histogram.xbins

go_histogram_ybins() histogram.ybins

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_histogram(x = 1:3, y = c(2, 1, 3), marker_color = "red")
```

---

add_histogram2d *Histogram2d traces*

Description

The sample data from which statistics are computed is set in 'x' and 'y' (where 'x' and 'y' represent marginal distributions, binning is set in 'xbins' and 'ybins' in this case) or 'z' (where 'z' represent the 2D distribution and binning set, binning is set by 'x' and 'y' in this case). The resulting distribution is visualized as a heatmap.

Usage

```
add_histogram2d(fig, ...)

## Default S3 method:
add_histogram2d(
  fig,
  ...,
  data = NULL,
  autobinx = NULL,
  autobiny = NULL,
  autocolorscale = NULL,
  bingroup = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  customdata = NULL,
  histfunc = NULL,
  histnorm = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  nbinsx = NULL,
  nbinsy = NULL,
  opacity = NULL,
  reversescale = NULL,
  showlegend = NULL,
  showscale = NULL,
  textfont = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  xaxis = NULL,
  xbingroup = NULL,
  xbins = NULL,
  xcalendar = NULL,
```

```
  xgap = NULL,
  xhoverformat = NULL,
  y = NULL,
  yaxis = NULL,
  ybingroup = NULL,
  ybins = NULL,
  ycalendar = NULL,
  ygap = NULL,
  yhoverformat = NULL,
  z = NULL,
  zauto = NULL,
  zhoverformat = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  zsmooth = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_histogram2d(
  arg = NULL,
  ...,
  autobinx = NULL,
  autobiny = NULL,
  autocolorscale = NULL,
  bingroup = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  customdata = NULL,
  histfunc = NULL,
  histnorm = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
```

```
  nbinsx = NULL,
  nbinsy = NULL,
  opacity = NULL,
  reversescale = NULL,
  showlegend = NULL,
  showscale = NULL,
  textfont = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  xaxis = NULL,
  xbingroup = NULL,
  xbins = NULL,
  xcalendar = NULL,
  xgap = NULL,
  xhoverformat = NULL,
  y = NULL,
  yaxis = NULL,
  ybingroup = NULL,
  ybins = NULL,
  ycalendar = NULL,
  ygap = NULL,
  yhoverformat = NULL,
  z = NULL,
  zauto = NULL,
  zhoverformat = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  zsmooth = NULL,
  skip_invalid = FALSE
)

go_histogram2d_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
```

```
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_histogram2d_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
```

```
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram2d_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_histogram2d_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_histogram2d_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram2d_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
```

```
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_histogram2d_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram2d_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_histogram2d_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram2d_marker(arg = NULL, ..., color = NULL, skip_invalid = FALSE)

go_histogram2d_textfont(
  arg = NULL,
```

```
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram2d_xbins(
  arg = NULL,
  ...,
  end = NULL,
  size = NULL,
  start = NULL,
  skip_invalid = FALSE
)

go_histogram2d_ybins(
  arg = NULL,
  ...,
  end = NULL,
  size = NULL,
  start = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
autobinx	Obsolete: since v1.42 each bin attribute is auto-determined separately and 'autobinx' is not needed. However, we accept 'autobinx: true' or 'false' and will update 'xbins' accordingly before deleting 'autobinx' from the trace. Logical.
autobiny	Obsolete: since v1.42 each bin attribute is auto-determined separately and 'autobiny' is not needed. However, we accept 'autobiny: true' or 'false' and will update 'ybins' accordingly before deleting 'autobiny' from the trace. Logical.

`autocolorscale` Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`bingroup` Set the ‘xbingroup‘ and ‘ybingroup‘ default prefix For example, setting a ‘bingroup‘ of 1 on two histogram2d traces will make them their x-bins and y-bins match separately. String.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_histogram2d_colorbar()` object.

`colorscale` Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘zmin‘ and ‘zmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`customdata` Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`histfunc` Specifies the binning function used for this histogram trace. If "count", the histogram values are computed by counting the number of values lying inside each bin. If "sum", "avg", "min", "max", the histogram values are computed using the sum, the average, the minimum or the maximum of the values lying inside each bin respectively. One of `"count"`, `"sum"`, `"avg"`, `"min"`, `"max"`.

`histnorm` Specifies the type of normalization used for this histogram trace. If *", the span of each bar corresponds to the number of occurrences (i.e. the number of data points lying inside the bins). If "percent" / "probability", the span of each bar corresponds to the percentage / fraction of occurrences with respect to the total number of sample points (here, the sum of all bin HEIGHTS equals 100% / 1). If "density", the span of each bar corresponds to the number of occurrences in a bin divided by the size of the bin interval (here, the sum of all bin AREAS equals the total number of sample points). If "probability density*, the area of each bar corresponds to the probability that an event will fall into the corresponding bin (here, the sum of all bin AREAS equals 1). One of `""`, `"percent"`, `"probability"`, `"density"`, `"probability density"`.

`hoverinfo` Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_histogram2d_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variable 'z' Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle` Named list or `go_histogram2d_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`marker` Named list or `go_histogram2d_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_histogram2d()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_histogram2d_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`nbinsx` Specifies the maximum number of desired bins. This value will be used in an algorithm that will decide the optimal bin size such that the histogram best visualizes the distribution of the data. Ignored if ‘xbins.size‘ is provided. Integer in the interval [0, Inf].

`nbinsy` Specifies the maximum number of desired bins. This value will be used in an algorithm that will decide the optimal bin size such that the histogram best visualizes the distribution of the data. Ignored if ‘ybins.size‘ is provided. Integer in the interval [0, Inf].

`opacity` Sets the opacity of the trace. Number in the interval [0, 1].

`reversescale` Reverses the color mapping if true. If true, ‘zmin‘ will correspond to the last color in the array and ‘zmax‘ will correspond to the first color. Logical.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Logical.

`textfont` Sets the text font. Named list or `go_histogram2d_textfont()` object.

`texttemplate` Template string used for rendering the information text that appears on points. Note that this will override ‘textinfo‘. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variable ‘z‘ String.

texttemplatefallback
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

uid
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision
: Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

visible
: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

x
: `go_histogram2d()`: Sets the sample data to be binned on the x axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  `go_histogram2d_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

xaxis
: Sets a reference between this trace's x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

xbingroup
: Set a group of histogram traces which will have compatible x-bin settings. Using ‘xbingroup‘, histogram2d and histogram2dcontour traces (on axes of the same axis type) can have compatible x-bin settings. Note that the same ‘xbingroup‘ value can be used to set (1D) histogram ‘bingroup‘ String.

xbins
: Named list or `go_histogram2d_xbins()` object.

xcalendar
: Sets the calendar system to use with ‘x‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

xgap
: Sets the horizontal gap (in pixels) between bricks. Number in the interval [0, Inf].

xhoverformat
: Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter:

"%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

`y` `go_histogram2d()`: Sets the sample data to be binned on the y axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_histogram2d_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

`yaxis` Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...

`ybingroup` Set a group of histogram traces which will have compatible y-bin settings. Using ‘ybingroup‘, histogram2d and histogram2dcontour traces (on axes of the same axis type) can have compatible y-bin settings. Note that the same ‘ybingroup‘ value can be used to set (1D) histogram ‘bingroup‘ String.

`ybins` Named list or `go_histogram2d_ybins()` object.

`ycalendar` Sets the calendar system to use with ‘y‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

`ygap` Sets the vertical gap (in pixels) between bricks. Number in the interval [0, Inf].

`yhoverformat` Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

`z` Sets the aggregation data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`zauto` Determines whether or not the color domain is computed with respect to the input data (here in ‘z‘) or the bounds set in ‘zmin‘ and ‘zmax‘ Defaults to ‘false‘ when ‘zmin‘ and ‘zmax‘ are set by the user. Logical.

`zhoverformat` Sets the hover text formatting rule for ‘z‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

`zmax` Sets the upper bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmin‘ must be set as well. Number.

zmid	Sets the mid-point of the color domain by scaling ‘zmin‘ and/or ‘zmax‘ to be equidistant to this point. Value should have the same units as in ‘z‘. Has no effect when ‘zauto‘ is ‘false‘. Number.
zmin	Sets the lower bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmax‘ must be set as well. Number.
zsmooth	Picks a smoothing algorithm use to smooth ‘z‘ data. One of "fast", "best", FALSE.
row	Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
secondary_y	If TRUE, use the secondary y axis of the subplot.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
bgcolor	go_histogram2d_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name. go_histogram2d_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.
bordercolor	go_histogram2d_colorbar(): Sets the axis line color. Color: a CSS color string or R color name. go_histogram2d_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.
borderwidth	Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].
dtick	Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.
exponentformat	Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".
labelalias	Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_histogram2d_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops` List of named lists or `go_histogram2d_colorbar_tickformatstop()` objects.

`ticklabeloverflow` Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition` Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

title	String (the title text), named list or go_histogram2d_colorbar_title() object.
xanchor	Sets this color bar's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the color bar. Defaults to "left" when 'orientation' is "v" and "center" when 'orientation' is "h". One of "left", "center", "right".
xpad	Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].
xref	Sets the container 'x' refers to. "container" spans the entire 'width' of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".
yanchor	Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of "top", "middle", "bottom".
ypad	Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].
yref	Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".
color	go_histogram2d_colorbar_tickfont(), go_histogram2d_colorbar_title_font(), go_histogram2d_legendgrouptitle_font(), go_histogram2d_textfont(): Color: a CSS color string or R color name. go_histogram2d_hoverlabel_font(): Color: a CSS color string or R color name, or a vector of these. go_histogram2d_marker(): Sets the aggregation data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
family	go_histogram2d_colorbar_tickfont(), go_histogram2d_colorbar_title_font(), go_histogram2d_legendgrouptitle_font(), go_histogram2d_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String. go_histogram2d_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.
lineposition	go_histogram2d_colorbar_tickfont(), go_histogram2d_colorbar_title_font(), go_histogram2d_legendgrouptitle_font(), go_histogram2d_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none". go_histogram2d_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

shadow

go_histogram2d_colorbar_tickfont(), go_histogram2d_colorbar_title_font(), go_histogram2d_legendgrouptitle_font(), go_histogram2d_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

go_histogram2d_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size

go_histogram2d_colorbar_tickfont(), go_histogram2d_colorbar_title_font(), go_histogram2d_legendgrouptitle_font(), go_histogram2d_textfont(): Number in the interval [1, Inf].

go_histogram2d_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

go_histogram2d_xbins(): Sets the size of each x axis bin. Default behavior: If ‘nbinsx‘ is 0 or omitted, we choose a nice round bin size such that the number of bins is about the same as the typical number of samples in each bin. If ‘nbinsx‘ is provided, we choose a nice round bin size giving no more than that many bins. For date data, use milliseconds or "M<n>" for months, as in ‘axis.dtick‘. For category data, the number of categories to bin together (always defaults to 1). Any value.

go_histogram2d_ybins(): Sets the size of each y axis bin. Default behavior: If ‘nbinsy‘ is 0 or omitted, we choose a nice round bin size such that the number of bins is about the same as the typical number of samples in each bin. If ‘nbinsy‘ is provided, we choose a nice round bin size giving no more than that many bins. For date data, use milliseconds or "M<n>" for months, as in ‘axis.dtick‘. For category data, the number of categories to bin together (always defaults to 1). Any value.

style

go_histogram2d_colorbar_tickfont(), go_histogram2d_colorbar_title_font(), go_histogram2d_legendgrouptitle_font(), go_histogram2d_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

go_histogram2d_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

textcase

go_histogram2d_colorbar_tickfont(), go_histogram2d_colorbar_title_font(), go_histogram2d_legendgrouptitle_font(), go_histogram2d_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

go_histogram2d_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant

go_histogram2d_colorbar_tickfont(), go_histogram2d_colorbar_title_font(), go_histogram2d_legendgrouptitle_font(), go_histogram2d_textfont():

Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

go_histogram2d_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight — go_histogram2d_colorbar_tickfont(), go_histogram2d_colorbar_title_font(), go_histogram2d_legendgrouptitle_font(), go_histogram2d_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

go_histogram2d_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

font — go_histogram2d_colorbar_title(): Sets this color bar's title font. Named list or go_histogram2d_colorbar_title_font() object.

go_histogram2d_hoverlabel(): Sets the font used in hover labels. Named list or go_histogram2d_hoverlabel_font() object.

go_histogram2d_legendgrouptitle(): Sets this legend group's title font. Named list or go_histogram2d_legendgrouptitle_font() object.

side — Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of "right", "top", "bottom".

text — go_histogram2d_colorbar_title(): Sets the title of the color bar. String.

go_histogram2d_legendgrouptitle(): Sets the title of the legend group. String.

align — Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.

namelength — Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

end
: go_histogram2d_xbins(): Sets the end value for the x axis bins. The last bin may not end exactly at this value, we increment the bin edge by 'size' from 'start' until we reach or exceed 'end'. Defaults to the maximum data value. Like 'start', for dates use a date string, and for category data 'end' is based on the category serial numbers. Any value.

: go_histogram2d_ybins(): Sets the end value for the y axis bins. The last bin may not end exactly at this value, we increment the bin edge by 'size' from 'start' until we reach or exceed 'end'. Defaults to the maximum data value. Like 'start', for dates use a date string, and for category data 'end' is based on the category serial numbers. Any value.

start
: go_histogram2d_xbins(): Sets the starting value for the x axis bins. Defaults to the minimum data value, shifted down if necessary to make nice round values and to remove ambiguous bin edges. For example, if most of the data is integers we shift the bin edges 0.5 down, so a 'size' of 5 would have a default 'start' of -0.5, so it is clear that 0-4 are in the first bin, 5-9 in the second, but continuous data gets a start of 0 and bins [0,5), [5,10) etc. Dates behave similarly, and 'start' should be a date string. For category data, 'start' is based on the category serial numbers, and defaults to -0.5. Any value.

: go_histogram2d_ybins(): Sets the starting value for the y axis bins. Defaults to the minimum data value, shifted down if necessary to make nice round values and to remove ambiguous bin edges. For example, if most of the data is integers we shift the bin edges 0.5 down, so a 'size' of 5 would have a default 'start' of -0.5, so it is clear that 0-4 are in the first bin, 5-9 in the second, but continuous data gets a start of 0 and bins [0,5), [5,10) etc. Dates behave similarly, and 'start' should be a date string. For category data, 'start' is based on the category serial numbers, and defaults to -0.5. Any value.

Details

go_histogram2d() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_histogram2d() creates an object and adds it to a figure.

Nested objects documented here:

go_histogram2d_colorbar() histogram2d.colorbar

go_histogram2d_colorbar_tickfont() histogram2d.colorbar.tickfont: Sets the color bar's tick label font

go_histogram2d_colorbar_tickformatstop() histogram2d.colorbar.tickformatstop

go_histogram2d_colorbar_title() histogram2d.colorbar.title

go_histogram2d_colorbar_title_font() histogram2d.colorbar.title.font: Sets this color bar's title font.

go_histogram2d_hoverlabel() histogram2d.hoverlabel

go_histogram2d_hoverlabel_font() histogram2d.hoverlabel.font: Sets the font used in hover labels.

`go_histogram2d_legendgrouptitle()` `histogram2d.legendgrouptitle`

`go_histogram2d_legendgrouptitle_font()` `histogram2d.legendgrouptitle.font`: Sets this legend group's title font.

`go_histogram2d_marker()` `histogram2d.marker`

`go_histogram2d_textfont()` `histogram2d.textfont`: Sets the text font.

`go_histogram2d_xbins()` `histogram2d.xbins`

`go_histogram2d_ybins()` `histogram2d.ybins`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_histogram2d(x = 1:3, y = c(2, 1, 3), marker_color = "red")
```

---

add_histogram2dcontour

Histogram2dcontour traces

Description

The sample data from which statistics are computed is set in 'x' and 'y' (where 'x' and 'y' represent marginal distributions, binning is set in 'xbins' and 'ybins' in this case) or 'z' (where 'z' represent the 2D distribution and binning set, binning is set by 'x' and 'y' in this case). The resulting distribution is visualized as a contour plot.

Usage

```
add_histogram2dcontour(fig, ...)

## Default S3 method:
add_histogram2dcontour(
  fig,
  ...,
  data = NULL,
  autobinx = NULL,
  autobiny = NULL,
  autocolorscale = NULL,
  autocontour = NULL,
  bingroup = NULL,
```

```
    coloraxis = NULL,
    colorbar = NULL,
    colorscale = NULL,
    contours = NULL,
    customdata = NULL,
    histfunc = NULL,
    histnorm = NULL,
    hoverinfo = NULL,
    hoverlabel = NULL,
    hovertemplate = NULL,
    hovertemplatefallback = NULL,
    ids = NULL,
    legend = NULL,
    legendgroup = NULL,
    legendgrouptitle = NULL,
    legendrank = NULL,
    legendwidth = NULL,
    line = NULL,
    marker = NULL,
    meta = NULL,
    name = NULL,
    nbinsx = NULL,
    nbinsy = NULL,
    ncontours = NULL,
    opacity = NULL,
    reversescale = NULL,
    showlegend = NULL,
    showscale = NULL,
    textfont = NULL,
    texttemplate = NULL,
    texttemplatefallback = NULL,
    uid = NULL,
    uirevision = NULL,
    visible = NULL,
    x = NULL,
    xaxis = NULL,
    xbingroup = NULL,
    xbins = NULL,
    xcalendar = NULL,
    xhoverformat = NULL,
    y = NULL,
    yaxis = NULL,
    ybingroup = NULL,
    ybins = NULL,
    ycalendar = NULL,
    yhoverformat = NULL,
    z = NULL,
    zauto = NULL,
```

```
  zhoverformat = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_histogram2dcontour(
  arg = NULL,
  ...,
  autobinx = NULL,
  autobiny = NULL,
  autocolorscale = NULL,
  autocontour = NULL,
  bingroup = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  contours = NULL,
  customdata = NULL,
  histfunc = NULL,
  histnorm = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  nbinsx = NULL,
  nbinsy = NULL,
  ncontours = NULL,
  opacity = NULL,
  reversescale = NULL,
  showlegend = NULL,
  showscale = NULL,
  textfont = NULL,
  texttemplate = NULL,
```

```
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  xaxis = NULL,
  xbingroup = NULL,
  xbins = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  y = NULL,
  yaxis = NULL,
  ybingroup = NULL,
  ybins = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  z = NULL,
  zauto = NULL,
  zhoverformat = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  skip_invalid = FALSE
)

go_histogram2dcontour_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
```

```
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_histogram2dcontour_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram2dcontour_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
```

```
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_histogram2dcontour_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_histogram2dcontour_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram2dcontour_contours(
  arg = NULL,
  ...,
  coloring = NULL,
  end = NULL,
  labelfont = NULL,
  labelformat = NULL,
  operation = NULL,
  showlabels = NULL,
  showlines = NULL,
  size = NULL,
  start = NULL,
  type = NULL,
  value = NULL,
  skip_invalid = FALSE
)
```

```
  go_histogram2dcontour_contours_labelfont(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_histogram2dcontour_hoverlabel(
    arg = NULL,
    ...,
    align = NULL,
    bgcolor = NULL,
    bordercolor = NULL,
    font = NULL,
    namelength = NULL,
    showarrow = NULL,
    skip_invalid = FALSE
  )

  go_histogram2dcontour_hoverlabel_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_histogram2dcontour_legendgrouptitle(
    arg = NULL,
    ...,
    font = NULL,
    text = NULL,
    skip_invalid = FALSE
```

```
)

go_histogram2dcontour_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_histogram2dcontour_line(
  arg = NULL,
  ...,
  color = NULL,
  dash = NULL,
  smoothing = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_histogram2dcontour_marker(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

go_histogram2dcontour_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

```
go_histogram2dcontour_xbins(
  arg = NULL,
  ...,
  end = NULL,
  size = NULL,
  start = NULL,
  skip_invalid = FALSE
)

go_histogram2dcontour_ybins(
  arg = NULL,
  ...,
  end = NULL,
  size = NULL,
  start = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
autobinx	Obsolete: since v1.42 each bin attribute is auto-determined separately and 'autobinx' is not needed. However, we accept 'autobinx: true' or 'false' and will update 'xbins' accordingly before deleting 'autobinx' from the trace. Logical.
autobiny	Obsolete: since v1.42 each bin attribute is auto-determined separately and 'autobiny' is not needed. However, we accept 'autobiny: true' or 'false' and will update 'ybins' accordingly before deleting 'autobiny' from the trace. Logical.
autocolorscale	Determines whether the colorscale is a default palette ('autocolorscale: true') or the palette determined by 'colorscale'. In case 'colorscale' is unspecified or 'autocolorscale' is true, the default palette will be chosen according to whether numbers in the 'color' array are all positive, all negative or mixed. Logical.
autocontour	Determines whether or not the contour level attributes are picked by an algorithm. If TRUE, the number of contour levels can be set in 'ncontours'. If FALSE, set the contour level attributes in 'contours'. Logical.
bingroup	Set the 'xbingroup' and 'ybingroup' default prefix For example, setting a 'bingroup' of 1 on two histogram2d traces will make them their x-bins and y-bins match separately. String.
coloraxis	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes

are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: "coloraxis", "coloraxis2", "coloraxis3", ...

colorbar — Named list or go_histogram2dcontour_colorbar() object.

colorscale — Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘zmin‘ and ‘zmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

contours — Named list or go_histogram2dcontour_contours() object.

customdata — Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

histfunc — Specifies the binning function used for this histogram trace. If "count", the histogram values are computed by counting the number of values lying inside each bin. If "sum", "avg", "min", "max", the histogram values are computed using the sum, the average, the minimum or the maximum of the values lying inside each bin respectively. One of "count", "sum", "avg", "min", "max".

histnorm — Specifies the type of normalization used for this histogram trace. If *", the span of each bar corresponds to the number of occurrences (i.e. the number of data points lying inside the bins). If "percent" / "probability", the span of each bar corresponds to the percentage / fraction of occurrences with respect to the total number of sample points (here, the sum of all bin HEIGHTS equals 100% / 1). If "density", the span of each bar corresponds to the number of occurrences in a bin divided by the size of the bin interval (here, the sum of all bin AREAS equals the total number of sample points). If "probability density*, the area of each bar corresponds to the probability that an event will fall into the corresponding bin (here, the sum of all bin AREAS equals 1). One of "", "percent", "probability", "density", "probability density".

hoverinfo — Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of "x", "y", "z", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.

hoverlabel — Named list or go_histogram2dcontour_hoverlabel() object.

hovertemplate — Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown.

Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variable 'z' Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_histogram2dcontour_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line`
: Named list or `go_histogram2dcontour_line()` object.

`marker`
: Named list or `go_histogram2dcontour_marker()` object.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute

in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_histogram2dcontour()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_histogram2dcontour_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`nbinsx` Specifies the maximum number of desired bins. This value will be used in an algorithm that will decide the optimal bin size such that the histogram best visualizes the distribution of the data. Ignored if ‘xbins.size‘ is provided. Integer in the interval [0, Inf].

`nbinsy` Specifies the maximum number of desired bins. This value will be used in an algorithm that will decide the optimal bin size such that the histogram best visualizes the distribution of the data. Ignored if ‘ybins.size‘ is provided. Integer in the interval [0, Inf].

`ncontours` Sets the maximum number of contour levels. The actual number of contours will be chosen automatically to be less than or equal to the value of ‘ncontours‘. Has an effect only if ‘autocontour‘ is TRUE or if ‘contours.size‘ is missing. Integer in the interval [1, Inf].

`opacity` Sets the opacity of the trace. Number in the interval [0, 1].

`reversescale` Reverses the color mapping if true. If true, ‘zmin‘ will correspond to the last color in the array and ‘zmax‘ will correspond to the first color. Logical.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Logical.

`textfont` For this trace it only has an effect if ‘contours.coloring‘ is set to "heatmap". Sets the text font. Named list or `go_histogram2dcontour_textfont()` object.

`texttemplate` For this trace it only has an effect if ‘contours.coloring‘ is set to "heatmap". Template string used for rendering the information text that appears on points. Note that this will override ‘textinfo‘. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variables ‘x‘, ‘y‘, ‘z‘ and ‘text‘. String.

texttemplatefallback
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

uid
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision
: Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

visible
: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

x
: `go_histogram2dcontour()`: Sets the sample data to be binned on the x axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  `go_histogram2dcontour_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

xaxis
: Sets a reference between this trace's x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

xbingroup
: Set a group of histogram traces which will have compatible x-bin settings. Using ‘xbingroup‘, histogram2d and histogram2dcontour traces (on axes of the same axis type) can have compatible x-bin settings. Note that the same ‘xbingroup‘ value can be used to set (1D) histogram ‘bingroup‘ String.

xbins
: Named list or `go_histogram2dcontour_xbins()` object.

xcalendar
: Sets the calendar system to use with ‘x‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

xhoverformat
: Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional

seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

`y` `go_histogram2dcontour()`: Sets the sample data to be binned on the y axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_histogram2dcontour_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

`yaxis` Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...

`ybingroup` Set a group of histogram traces which will have compatible y-bin settings. Using ‘ybingroup‘, histogram2d and histogram2dcontour traces (on axes of the same axis type) can have compatible y-bin settings. Note that the same ‘ybingroup‘ value can be used to set (1D) histogram ‘bingroup‘ String.

`ybins` Named list or `go_histogram2dcontour_ybins()` object.

`ycalendar` Sets the calendar system to use with ‘y‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

`yhoverformat` Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

`z` Sets the aggregation data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`zauto` Determines whether or not the color domain is computed with respect to the input data (here in ‘z‘) or the bounds set in ‘zmin‘ and ‘zmax‘ Defaults to ‘false‘ when ‘zmin‘ and ‘zmax‘ are set by the user. Logical.

`zhoverformat` Sets the hover text formatting rule for ‘z‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

`zmax` Sets the upper bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmin‘ must be set as well. Number.

`zmid` Sets the mid-point of the color domain by scaling ‘zmin‘ and/or ‘zmax‘ to be equidistant to this point. Value should have the same units as in ‘z‘. Has no effect when ‘zauto‘ is ‘false‘. Number.

zmin	Sets the lower bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmax‘ must be set as well. Number.
row	Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
secondary_y	If TRUE, use the secondary y axis of the subplot.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
bgcolor	go_histogram2dcontour_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name. go_histogram2dcontour_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.
bordercolor	go_histogram2dcontour_colorbar(): Sets the axis line color. Color: a CSS color string or R color name. go_histogram2dcontour_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.
borderwidth	Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].
dtick	Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.
exponentformat	Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".
labelalias	Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.
len	Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_histogram2dcontour_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter:

"%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops`
: List of named lists or `go_histogram2dcontour_colorbar_tickformatstop()` objects.

`ticklabeloverflow`
: Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition`
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep`
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

`ticklen`
: Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode`
: Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix`
: Sets a tick label prefix. String.

`ticks`
: Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix`
: Sets a tick label suffix. String.

`ticktext`
: Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals`
: Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth`
: Sets the tick width (in px). Number in the interval [0, Inf].

`title`
: String (the title text), named list or `go_histogram2dcontour_colorbar_title()` object.

xanchor — Sets this color bar's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the color bar. Defaults to "left" when 'orientation' is "v" and "center" when 'orientation' is "h". One of `"left"`, `"center"`, `"right"`.

xpad — Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container 'x' refers to. "container" spans the entire 'width' of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

yanchor — Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of `"top"`, `"middle"`, `"bottom"`.

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

color — `go_histogram2dcontour_colorbar_tickfont()`, `go_histogram2dcontour_colorbar_title_font(`
`go_histogram2dcontour_contours_labelfont()`, `go_histogram2dcontour_legendgrouptitle_fo`
`go_histogram2dcontour_textfont()`: Color: a CSS color string or R color name.

`go_histogram2dcontour_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

`go_histogram2dcontour_line()`: Sets the color of the contour level. Has no effect if 'contours.coloring' is set to "lines". Color: a CSS color string or R color name.

`go_histogram2dcontour_marker()`: Sets the aggregation data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

family — `go_histogram2dcontour_colorbar_tickfont()`, `go_histogram2dcontour_colorbar_title_font(`
`go_histogram2dcontour_contours_labelfont()`, `go_histogram2dcontour_legendgrouptitle_fo`
`go_histogram2dcontour_textfont()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

`go_histogram2dcontour_hoverlabel_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

lineposition — `go_histogram2dcontour_colorbar_tickfont()`, `go_histogram2dcontour_colorbar_title_font(`
`go_histogram2dcontour_contours_labelfont()`, `go_histogram2dcontour_legendgrouptitle_fo`
`go_histogram2dcontour_textfont()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

go_histogram2dcontour_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

shadow

go_histogram2dcontour_colorbar_tickfont(), go_histogram2dcontour_colorbar_title_font(
go_histogram2dcontour_contours_labelfont(), go_histogram2dcontour_legendgrouptitle_fo
go_histogram2dcontour_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

go_histogram2dcontour_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size

go_histogram2dcontour_colorbar_tickfont(), go_histogram2dcontour_colorbar_title_font(
go_histogram2dcontour_contours_labelfont(), go_histogram2dcontour_legendgrouptitle_fo
go_histogram2dcontour_textfont(): Number in the interval [1, Inf].

go_histogram2dcontour_contours(): Sets the step between each contour level. Must be positive. Number in the interval [0, Inf].

go_histogram2dcontour_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

go_histogram2dcontour_xbins(): Sets the size of each x axis bin. Default behavior: If ‘nbinsx‘ is 0 or omitted, we choose a nice round bin size such that the number of bins is about the same as the typical number of samples in each bin. If ‘nbinsx‘ is provided, we choose a nice round bin size giving no more than that many bins. For date data, use milliseconds or "M<n>" for months, as in ‘axis.dtick‘. For category data, the number of categories to bin together (always defaults to 1). Any value.

go_histogram2dcontour_ybins(): Sets the size of each y axis bin. Default behavior: If ‘nbinsy‘ is 0 or omitted, we choose a nice round bin size such that the number of bins is about the same as the typical number of samples in each bin. If ‘nbinsy‘ is provided, we choose a nice round bin size giving no more than that many bins. For date data, use milliseconds or "M<n>" for months, as in ‘axis.dtick‘. For category data, the number of categories to bin together (always defaults to 1). Any value.

style

go_histogram2dcontour_colorbar_tickfont(), go_histogram2dcontour_colorbar_title_font(
go_histogram2dcontour_contours_labelfont(), go_histogram2dcontour_legendgrouptitle_fo
go_histogram2dcontour_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

go_histogram2dcontour_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

textcase

go_histogram2dcontour_colorbar_tickfont(), go_histogram2dcontour_colorbar_title_font(
go_histogram2dcontour_contours_labelfont(), go_histogram2dcontour_legendgrouptitle_fo
go_histogram2dcontour_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

go_histogram2dcontour_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant

go_histogram2dcontour_colorbar_tickfont(), go_histogram2dcontour_colorbar_title_font(go_histogram2dcontour_contours_labelfont(), go_histogram2dcontour_legendgrouptitle_fo go_histogram2dcontour_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

go_histogram2dcontour_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight

go_histogram2dcontour_colorbar_tickfont(), go_histogram2dcontour_colorbar_title_font(go_histogram2dcontour_contours_labelfont(), go_histogram2dcontour_legendgrouptitle_fo go_histogram2dcontour_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

go_histogram2dcontour_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange

range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled

Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

templateitemname

Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

value

go_histogram2dcontour_colorbar_tickformatstop(): string - dtickformat for described zoom level, the same as "tickformat" String.

go_histogram2dcontour_contours(): Sets the value or values of the constraint boundary. When 'operation' is set to one of the comparison values ('=,<,>=,>,<=') "value" is expected to be a number. When 'operation' is set to one of the interval values ('[],(),[),(],][,)(,](,)[') "value" is expected to be an array of two numbers where the first is the lower bound and the second is the upper bound. Any value.

font

go_histogram2dcontour_colorbar_title(): Sets this color bar's title font. Named list or go_histogram2dcontour_colorbar_title_font() object.

go_histogram2dcontour_hoverlabel(): Sets the font used in hover labels. Named list or go_histogram2dcontour_hoverlabel_font() object.

go_histogram2dcontour_legendgrouptitle(): Sets this legend group's title font. Named list or go_histogram2dcontour_legendgrouptitle_font() object.

side — Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of `"right"`, `"top"`, `"bottom"`.

text — `go_histogram2dcontour_colorbar_title()`: Sets the title of the color bar. String.

`go_histogram2dcontour_legendgrouptitle()`: Sets the title of the legend group. String.

coloring — Determines the coloring method showing the contour values. If "fill", coloring is done evenly between each contour level If "heatmap", a heatmap gradient coloring is applied between each contour level. If "lines", coloring is done on the contour lines. If "none", no coloring is applied on this trace. One of `"fill"`, `"heatmap"`, `"lines"`, `"none"`.

end — `go_histogram2dcontour_contours()`: Sets the end contour level value. Must be more than 'contours.start' Number.

`go_histogram2dcontour_xbins()`: Sets the end value for the x axis bins. The last bin may not end exactly at this value, we increment the bin edge by 'size' from 'start' until we reach or exceed 'end'. Defaults to the maximum data value. Like 'start', for dates use a date string, and for category data 'end' is based on the category serial numbers. Any value.

`go_histogram2dcontour_ybins()`: Sets the end value for the y axis bins. The last bin may not end exactly at this value, we increment the bin edge by 'size' from 'start' until we reach or exceed 'end'. Defaults to the maximum data value. Like 'start', for dates use a date string, and for category data 'end' is based on the category serial numbers. Any value.

labelfont — Sets the font used for labeling the contour levels. The default color comes from the lines, if shown. The default family and size come from 'layout.font'. Named list or `go_histogram2dcontour_contours_labelfont()` object.

labelformat — Sets the contour label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. String.

operation — Sets the constraint operation. "=" keeps regions equal to 'value' "<" and "<=" keep regions less than 'value' ">" and ">=" keep regions greater than 'value' "[]", "()", "[)", and "(]" keep regions inside 'value[0]' to 'value[1]' "][", ")(", "](", ")[" keep regions outside 'value[0]' to 'value[1]' Open vs. closed intervals make no difference to constraint display, but all versions are allowed. One of `"="`, `"<"`, `">="`, `">"`, `"<="`, `"[]"`, `"()"`, `"[)"`, `"(]"`, `"][", ")(", "](", ")[".`

showlabels — Determines whether to label the contour lines with their values. Logical.

showlines — Determines whether or not the contour lines are drawn. Has an effect only if 'contours.coloring' is set to "fill". Logical.

start — `go_histogram2dcontour_contours()`: Sets the starting contour level value. Must be less than 'contours.end' Number.

`go_histogram2dcontour_xbins()`: Sets the starting value for the x axis bins. Defaults to the minimum data value, shifted down if necessary to make nice round values and to remove ambiguous bin edges. For example, if most of the data is integers we shift the bin edges 0.5 down, so a 'size' of 5 would have a

default 'start' of -0.5, so it is clear that 0-4 are in the first bin, 5-9 in the second, but continuous data gets a start of 0 and bins [0,5), [5,10) etc. Dates behave similarly, and 'start' should be a date string. For category data, 'start' is based on the category serial numbers, and defaults to -0.5. Any value.

`go_histogram2dcontour_ybins()`: Sets the starting value for the y axis bins. Defaults to the minimum data value, shifted down if necessary to make nice round values and to remove ambiguous bin edges. For example, if most of the data is integers we shift the bin edges 0.5 down, so a 'size' of 5 would have a default 'start' of -0.5, so it is clear that 0-4 are in the first bin, 5-9 in the second, but continuous data gets a start of 0 and bins [0,5), [5,10) etc. Dates behave similarly, and 'start' should be a date string. For category data, 'start' is based on the category serial numbers, and defaults to -0.5. Any value.

`type` If 'levels', the data is represented as a contour plot with multiple levels displayed. If 'constraint', the data is represented as constraints with the invalid region shaded as specified by the 'operation' and 'value' parameters. One of `"levels"`, `"constraint"`.

`align` Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`namelength` Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

`showarrow` Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

`dash` Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

`smoothing` Sets the amount of smoothing for the contour lines, where 0 corresponds to no smoothing. Number in the interval [0, 1.3].

`width` Sets the contour line width in (in px) Number in the interval [0, Inf].

Details

`go_histogram2dcontour()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_histogram2dcontour()` creates an object and adds it to a figure.

Nested objects documented here:

`go_histogram2dcontour_colorbar()` `histogram2dcontour.colorbar`

`go_histogram2dcontour_colorbar_tickfont()` `histogram2dcontour.colorbar.tickfont`: Sets the color bar's tick label font

go_histogram2dcontour_colorbar_tickformatstop() histogram2dcontour.colorbar.tickformatstop

go_histogram2dcontour_colorbar_title() histogram2dcontour.colorbar.title

go_histogram2dcontour_colorbar_title_font() histogram2dcontour.colorbar.title.font: Sets this color bar's title font.

go_histogram2dcontour_contours() histogram2dcontour.contours

go_histogram2dcontour_contours_labelfont() histogram2dcontour.contours.labelfont: Sets the font used for labeling the contour levels. The default color comes from the lines, if shown. The default family and size come from 'layout.font'.

go_histogram2dcontour_hoverlabel() histogram2dcontour.hoverlabel

go_histogram2dcontour_hoverlabel_font() histogram2dcontour.hoverlabel.font: Sets the font used in hover labels.

go_histogram2dcontour_legendgrouptitle() histogram2dcontour.legendgrouptitle

go_histogram2dcontour_legendgrouptitle_font() histogram2dcontour.legendgrouptitle.font: Sets this legend group's title font.

go_histogram2dcontour_line() histogram2dcontour.line

go_histogram2dcontour_marker() histogram2dcontour.marker

go_histogram2dcontour_textfont() histogram2dcontour.textfont: For this trace it only has an effect if 'contours.coloring' is set to "heatmap". Sets the text font.

go_histogram2dcontour_xbins() histogram2dcontour.xbins

go_histogram2dcontour_ybins() histogram2dcontour.ybins

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_histogram2dcontour(x = 1:3, y = c(2, 1, 3), marker_color = "red")
```

---

add_icicle *Icicle traces*

Description

Visualize hierarchal data from leaves (and/or outer branches) towards root with rectangles. The icicle sectors are determined by the entries in "labels" or "ids" and in "parents".

Usage

```
add_icicle(fig, ...)

## Default S3 method:
add_icicle(
  fig,
  ...,
  data = NULL,
  branchvalues = NULL,
  count = NULL,
  customdata = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextfont = NULL,
  labels = NULL,
  leaf = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  level = NULL,
  marker = NULL,
  maxdepth = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  outsidetextfont = NULL,
  parents = NULL,
  pathbar = NULL,
  root = NULL,
  sort = NULL,
  text = NULL,
  textfont = NULL,
  textinfo = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  tiling = NULL,
  uid = NULL,
  uirevision = NULL,
  values = NULL,
  visible = NULL,
  row = NULL,
```

```
  col = NULL,
  skip_invalid = FALSE
)

go_icicle(
  arg = NULL,
  ...,
  branchvalues = NULL,
  count = NULL,
  customdata = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextfont = NULL,
  labels = NULL,
  leaf = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  level = NULL,
  marker = NULL,
  maxdepth = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  outsidetextfont = NULL,
  parents = NULL,
  pathbar = NULL,
  root = NULL,
  sort = NULL,
  text = NULL,
  textfont = NULL,
  textinfo = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  tiling = NULL,
  uid = NULL,
  uirevision = NULL,
  values = NULL,
  visible = NULL,
  skip_invalid = FALSE
)
```

```
  go_icicle_domain(
    arg = NULL,
    ...,
    column = NULL,
    row = NULL,
    x = NULL,
    y = NULL,
    skip_invalid = FALSE
  )

  go_icicle_hoverlabel(
    arg = NULL,
    ...,
    align = NULL,
    bgcolor = NULL,
    bordercolor = NULL,
    font = NULL,
    namelength = NULL,
    showarrow = NULL,
    skip_invalid = FALSE
  )

  go_icicle_hoverlabel_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_icicle_insidetextfont(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
```

```
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_icicle_leaf(arg = NULL, ..., opacity = NULL, skip_invalid = FALSE)

go_icicle_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_icicle_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_icicle_marker(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colors = NULL,
  colorscale = NULL,
  line = NULL,
  pattern = NULL,
  reversescale = NULL,
  showscale = NULL,
  skip_invalid = FALSE
)
```

```
  go_icicle_marker_colorbar(
    arg = NULL,
    ...,
    bgcolor = NULL,
    bordercolor = NULL,
    borderwidth = NULL,
    dtick = NULL,
    exponentformat = NULL,
    labelalias = NULL,
    len = NULL,
    lenmode = NULL,
    minexponent = NULL,
    nticks = NULL,
    orientation = NULL,
    outlinecolor = NULL,
    outlinewidth = NULL,
    separatethousands = NULL,
    showexponent = NULL,
    showticklabels = NULL,
    showtickprefix = NULL,
    showticksuffix = NULL,
    thickness = NULL,
    thicknessmode = NULL,
    tick0 = NULL,
    tickangle = NULL,
    tickcolor = NULL,
    tickfont = NULL,
    tickformat = NULL,
    tickformatstops = NULL,
    ticklabeloverflow = NULL,
    ticklabelposition = NULL,
    ticklabelstep = NULL,
    ticklen = NULL,
    tickmode = NULL,
    tickprefix = NULL,
    ticks = NULL,
    ticksuffix = NULL,
    ticktext = NULL,
    tickvals = NULL,
    tickwidth = NULL,
    title = NULL,
    x = NULL,
    xanchor = NULL,
    xpad = NULL,
    xref = NULL,
    y = NULL,
    yanchor = NULL,
```

```
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_icicle_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_icicle_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_icicle_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_icicle_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
```

```
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_icicle_marker_line(
    arg = NULL,
    ...,
    color = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_icicle_marker_pattern(
    arg = NULL,
    ...,
    bgcolor = NULL,
    fgcolor = NULL,
    fgopacity = NULL,
    fillmode = NULL,
    path = NULL,
    shape = NULL,
    size = NULL,
    solidity = NULL,
    skip_invalid = FALSE
  )

  go_icicle_outsidetextfont(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_icicle_pathbar(
    arg = NULL,
    ...,
    edgeshape = NULL,
```

```
  side = NULL,
  textfont = NULL,
  thickness = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_icicle_pathbar_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_icicle_root(arg = NULL, ..., color = NULL, skip_invalid = FALSE)

go_icicle_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_icicle_tiling(
  arg = NULL,
  ...,
  flip = NULL,
  orientation = NULL,
  pad = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
branchvalues	Determines how the items in ‘values‘ are summed. When set to "total", items in ‘values‘ are taken to be value of all its descendants. When set to "remainder", items in ‘values‘ corresponding to the root and the branches sectors are taken to be the extra part not part of the sum of the values at their leaves. One of "remainder", "total".
count	Determines default for ‘values‘ when it is not provided, by inferring a 1 for each of the "leaves" and/or "branches", otherwise 0. Any combination of "branches", "leaves" joined with "+".
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
domain	Named list or go_icicle_domain() object.
hoverinfo	Determines what trace information appears on hover. Flags are rendered in a fixed order; use ‘hovertemplate‘ if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of "label", "text", "value", "name", "current path", "percent root", "percent entry", "percent parent" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_icicle_hoverlabel() object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described

at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'currentPath', 'root', 'entry', 'percentRoot', 'percentEntry' and 'percentParent'. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each sector. If a single string, the same string appears for all data points. If an array of string, the items are mapped in order of this trace's sectors. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`insidetextfont`
: Sets the font used for 'textinfo' lying inside the sector. Named list or `go_icicle_insidetextfont()` object.

`labels`
: Sets the labels of each of the sectors. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`leaf`
: Named list or `go_icicle_leaf()` object.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgrouptitle`
: Named list or `go_icicle_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`level`
: Sets the level from which this trace hierarchy is rendered. Set 'level' to '"' to start from the root node in the hierarchy. Must be an "id" if 'ids' is filled in, otherwise plotly attempts to find a matching item in 'labels'. Any value.

`marker`
: Named list or `go_icicle_marker()` object.

`maxdepth`
: Sets the number of rendered sectors from any given 'level'. Set 'maxdepth' to "-1" to render all the levels in the hierarchy. Integer.

meta
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

name
: go_icicle(): Sets the trace name. The trace name appears as the legend item and on hover. String.

  go_icicle_marker_colorbar_tickformatstop(): When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

opacity
: go_icicle(): Sets the opacity of the trace. Number in the interval [0, 1].

  go_icicle_leaf(): Sets the opacity of the leaves. With colorscale it is defaulted to 1; otherwise it is defaulted to 0.7 Number in the interval [0, 1].

outsidetextfont
: Sets the font used for ‘textinfo‘ lying outside the sector. This option refers to the root of the hierarchy presented on top left corner of a treemap graph. Please note that if a hierarchy has multiple root nodes, this option won't have any effect and ‘insidetextfont‘ would be used. Named list or go_icicle_outsidetextfont() object.

parents
: Sets the parent sectors for each of the sectors. Empty string items ” are understood to reference the root node in the hierarchy. If ‘ids‘ is filled, ‘parents‘ items are understood to be "ids" themselves. When ‘ids‘ is not set, plotly attempts to find matching items in ‘labels‘, but beware they must be unique. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

pathbar
: Named list or go_icicle_pathbar() object.

root
: Named list or go_icicle_root() object.

sort
: Determines whether or not the sectors are reordered from largest to smallest. Logical.

text
: go_icicle(): Sets text elements associated with each sector. If trace ‘textinfo‘ contains a "text" flag, these elements will be seen on the chart. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  go_icicle_legendgrouptitle(): Sets the title of the legend group. String.

  go_icicle_marker_colorbar_title(): Sets the title of the color bar. String.

textfont
: go_icicle(): Sets the font used for ‘textinfo‘. Named list or go_icicle_textfont() object.

  go_icicle_pathbar(): Sets the font used inside ‘pathbar‘. Named list or go_icicle_pathbar_textfont() object.

`textinfo` Determines what trace information appears on the graph. Flags are rendered in a fixed order; use 'texttemplate' if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of "label", "text", "value", "current path", "percent root", "percent entry", "percent parent" joined with "+", or one of "none".

`textposition` Sets the positions of the 'text' elements. One of "top left", "top center", "top right", "middle left", "middle center", "middle right", "bottom left", "bottom center", "bottom right".

`texttemplate` Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'currentPath', 'root', 'entry', 'percentRoot', 'percentEntry', 'percentParent', 'label' and 'value'. String, or a vector of these.

`texttemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`tiling` Named list or `go_icicle_tiling()` object.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

`values` Sets the values associated with each of the sectors. Use with 'branchvalues' to determine how the values are summed. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`visible` `go_icicle()`: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.

	`go_icicle_pathbar()`: Determines if the path bar is drawn i.e. outside the trace ‘domain‘ and with one pixel gap. Logical.
`row`	`add_icicle()`: Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells. `go_icicle_domain()`: If there is a layout grid, use the domain for this row in the grid for this icicle trace. Integer in the interval [0, Inf].
`col`	Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.
`skip_invalid`	If `TRUE`, silently drop invalid properties instead of raising an error.
`arg`	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
`column`	If there is a layout grid, use the domain for this column in the grid for this icicle trace. Integer in the interval [0, Inf].
`x`	`go_icicle_domain()`: Sets the horizontal domain of this icicle trace (in plot fraction). Vector or list of 2 element(s). `go_icicle_marker_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.
`y`	`go_icicle_domain()`: Sets the vertical domain of this icicle trace (in plot fraction). Vector or list of 2 element(s). `go_icicle_marker_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.
`align`	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.
`bgcolor`	`go_icicle_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these. `go_icicle_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name. `go_icicle_marker_pattern()`: When there is no colorscale sets the color of background pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "overlay". Otherwise, defaults to a transparent background. Color: a CSS color string or R color name, or a vector of these.
`bordercolor`	`go_icicle_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these. `go_icicle_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

font
: go_icicle_hoverlabel(): Sets the font used in hover labels. Named list or go_icicle_hoverlabel_font() object.
go_icicle_legendgrouptitle(): Sets this legend group's title font. Named list or go_icicle_legendgrouptitle_font() object.
go_icicle_marker_colorbar_title(): Sets this color bar's title font. Named list or go_icicle_marker_colorbar_title_font() object.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color
: go_icicle_hoverlabel_font(), go_icicle_insidetextfont(), go_icicle_outsidetextfont(), go_icicle_pathbar_textfont(), go_icicle_textfont(): Color: a CSS color string or R color name, or a vector of these.
go_icicle_legendgrouptitle_font(), go_icicle_marker_colorbar_tickfont(), go_icicle_marker_colorbar_title_font(): Color: a CSS color string or R color name.
go_icicle_marker_line(): Sets the color of the line enclosing each sector. Defaults to the 'paper_bgcolor' value. Color: a CSS color string or R color name, or a vector of these.
go_icicle_root(): sets the color of the root node for a sunburst/treemap/icicle trace. this has no effect when a colorscale is used to set the markers. Color: a CSS color string or R color name.

family
: go_icicle_hoverlabel_font(), go_icicle_insidetextfont(), go_icicle_outsidetextfont(), go_icicle_pathbar_textfont(), go_icicle_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.
go_icicle_legendgrouptitle_font(), go_icicle_marker_colorbar_tickfont(), go_icicle_marker_colorbar_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: go_icicle_hoverlabel_font(), go_icicle_insidetextfont(), go_icicle_outsidetextfont(), go_icicle_pathbar_textfont(), go_icicle_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.
go_icicle_legendgrouptitle_font(), go_icicle_marker_colorbar_tickfont(), go_icicle_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g.

"under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow
go_icicle_hoverlabel_font(), go_icicle_insidetextfont(), go_icicle_outsidetextfont(), go_icicle_pathbar_textfont(), go_icicle_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_icicle_legendgrouptitle_font(), go_icicle_marker_colorbar_tickfont(), go_icicle_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
go_icicle_hoverlabel_font(), go_icicle_insidetextfont(), go_icicle_outsidetextfont(), go_icicle_pathbar_textfont(), go_icicle_textfont(): Number in the interval [1, Inf], or a vector of these.

go_icicle_legendgrouptitle_font(), go_icicle_marker_colorbar_tickfont(), go_icicle_marker_colorbar_title_font(): Number in the interval [1, Inf].

go_icicle_marker_pattern(): Sets the size of unit squares of the pattern fill in pixels, which corresponds to the interval of repetition of the pattern. Number in the interval [0, Inf], or a vector of these.

style
go_icicle_hoverlabel_font(), go_icicle_insidetextfont(), go_icicle_outsidetextfont(), go_icicle_pathbar_textfont(), go_icicle_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_icicle_legendgrouptitle_font(), go_icicle_marker_colorbar_tickfont(), go_icicle_marker_colorbar_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
go_icicle_hoverlabel_font(), go_icicle_insidetextfont(), go_icicle_outsidetextfont(), go_icicle_pathbar_textfont(), go_icicle_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_icicle_legendgrouptitle_font(), go_icicle_marker_colorbar_tickfont(), go_icicle_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
go_icicle_hoverlabel_font(), go_icicle_insidetextfont(), go_icicle_outsidetextfont(), go_icicle_pathbar_textfont(), go_icicle_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_icicle_legendgrouptitle_font(), go_icicle_marker_colorbar_tickfont(), go_icicle_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
go_icicle_hoverlabel_font(), go_icicle_insidetextfont(), go_icicle_outsidetextfont(), go_icicle_pathbar_textfont(), go_icicle_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_icicle_legendgrouptitle_font(), go_icicle_marker_colorbar_tickfont(), go_icicle_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

autocolorscale Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if colors is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

cauto Determines whether or not the color domain is computed with respect to the input data (here colors) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if colors is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

cmax Sets the upper bound of the color domain. Has an effect only if colors is set to a numerical array. Value should have the same units as colors and if set, ‘marker.cmin‘ must be set as well. Number.

cmid Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if colors is set to a numerical array. Value should have the same units as colors. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

cmin Sets the lower bound of the color domain. Has an effect only if colors is set to a numerical array. Value should have the same units as colors and if set, ‘marker.cmax‘ must be set as well. Number.

coloraxis Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: "coloraxis", "coloraxis2", "coloraxis3", ...

colorbar Named list or go_icicle_marker_colorbar() object.

colors Sets the color of each sector of this trace. If not specified, the default trace color set is used to pick the sector colors. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

colorscale Sets the colorscale. Has an effect only if colors is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

line Named list or go_icicle_marker_line() object.

pattern Sets the pattern within the marker. Named list or go_icicle_marker_pattern() object.

`reversescale` Reverses the color mapping if true. Has an effect only if colors is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if colors is set to a numerical array. Logical.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` `go_icicle_marker_colorbar()`: Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`go_icicle_tiling()`: When set in conjunction with ‘tiling.flip‘, determines on which side the root nodes are drawn in the chart. If ‘tiling.orientation‘ is "v" and ‘tiling.flip‘ is *", the root nodes appear at the top. If ‘tiling.orientation‘ is "v" and ‘tiling.flip‘ is "y", the root nodes appear at the bottom. If ‘tiling.orientation‘ is "h" and ‘tiling.flip‘ is "*, the root nodes appear at the left. If ‘tiling.orientation‘ is "h" and ‘tiling.flip‘ is "x", the root nodes appear at the right. One of `"v"`, `"h"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` `go_icicle_marker_colorbar()`: Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`go_icicle_pathbar()`: Sets the thickness of ‘pathbar‘ (in px). If not specified the ‘pathbar.textfont.size‘ is used with 3 pixles extra padding on each side. Number in the interval [12, Inf].

`thicknessmode` Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar’s tick label font Named list or `go_icicle_marker_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops` List of named lists or `go_icicle_marker_colorbar_tickformatstop()` objects.

ticklabeloverflow
: Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelstep
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

ticklen
: Sets the tick length (in px). Number in the interval [0, Inf].

tickmode
: Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of "auto", "linear", "array".

tickprefix
: Sets a tick label prefix. String.

ticks
: Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of "outside", "inside", "".

ticksuffix
: Sets a tick label suffix. String.

ticktext
: Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals
: Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth
: Sets the tick width (in px). Number in the interval [0, Inf].

title
: String (the title text), named list or go_icicle_marker_colorbar_title() object.

xanchor
: Sets this color bar's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the color bar. Defaults to "left" when 'orientation' is "v" and "center" when 'orientation' is "h". One of "left", "center", "right".

xpad
: Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".

yanchor — Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

side — go_icicle_marker_colorbar_title(): Determines the location of color bar’s title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of "right", "top", "bottom".

go_icicle_pathbar(): Determines on which side of the treemap the ‘pathbar‘ should be presented. One of "top", "bottom".

width — Sets the width (in px) of the line enclosing each sector. Number in the interval [0, Inf], or a vector of these.

fgcolor — When there is no colorscale sets the color of foreground pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "replace". Otherwise, defaults to dark grey or white to increase contrast with the ‘bgcolor‘. Color: a CSS color string or R color name, or a vector of these.

fgopacity — Sets the opacity of the foreground pattern fill. Defaults to a 0.5 when ‘fillmode‘ is "overlay". Otherwise, defaults to 1. Number in the interval [0, 1].

fillmode — Determines whether ‘marker.color‘ should be used as a default to ‘bgcolor‘ or a ‘fgcolor‘. One of "replace", "overlay".

path — Sets a custom path for pattern fill. Use with no ‘shape‘ or ‘solidity‘, provide an SVG path string for the regions of the square from (0,0) to (‘size‘,‘size‘) to color. String, or a vector of these.

shape — Sets the shape of the pattern fill. By default, no pattern is used for filling the area. One of "", "/", "\\", "x", "-", "|", "+", ".", or a vector of these.

solidity	Sets the solidity of the pattern fill. Solidity is roughly the fraction of the area filled by the pattern. Solidity of 0 shows only the background color without pattern and solidty of 1 shows only the foreground color without pattern. Number in the interval [0, 1], or a vector of these.
edgeshape	Determines which shape is used for edges between 'barpath' labels. One of ">", "<", "|", "/", "\\".
flip	Determines if the positions obtained from solver are flipped on each axis. Any combination of "x", "y" joined with "+".
pad	Sets the inner padding (in px). Number in the interval [0, Inf].

Details

go_icicle() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_icicle() creates an object and adds it to a figure.

Nested objects documented here:

go_icicle_domain() icicle.domain

go_icicle_hoverlabel() icicle.hoverlabel

go_icicle_hoverlabel_font() icicle.hoverlabel.font: Sets the font used in hover labels.

go_icicle_insidetextfont() icicle.insidetextfont: Sets the font used for 'textinfo' lying inside the sector.

go_icicle_leaf() icicle.leaf

go_icicle_legendgrouptitle() icicle.legendgrouptitle

go_icicle_legendgrouptitle_font() icicle.legendgrouptitle.font: Sets this legend group's title font.

go_icicle_marker() icicle.marker

go_icicle_marker_colorbar() icicle.marker.colorbar

go_icicle_marker_colorbar_tickfont() icicle.marker.colorbar.tickfont: Sets the color bar's tick label font

go_icicle_marker_colorbar_tickformatstop() icicle.marker.colorbar.tickformatstop

go_icicle_marker_colorbar_title() icicle.marker.colorbar.title

go_icicle_marker_colorbar_title_font() icicle.marker.colorbar.title.font: Sets this color bar's title font.

go_icicle_marker_line() icicle.marker.line

go_icicle_marker_pattern() icicle.marker.pattern: Sets the pattern within the marker.

go_icicle_outsidetextfont() icicle.outsidetextfont: Sets the font used for 'textinfo' lying outside the sector. This option refers to the root of the hierarchy presented on top left corner of a treemap graph. Please note that if a hierarchy has multiple root nodes, this option won't have any effect and 'insidetextfont' would be used.

go_icicle_pathbar() icicle.pathbar

go_icicle_pathbar_textfont() icicle.pathbar.textfont: Sets the font used inside 'pathbar'.

`go_icicle_root()` icicle.root

`go_icicle_textfont()` icicle.textfont: Sets the font used for 'textinfo'.

`go_icicle_tiling()` icicle.tiling

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_icicle()
```

---

add_image *Image traces*

Description

Display an image, i.e. data on a 2D regular raster. By default, when an image is displayed in a subplot, its y axis will be reversed (ie. 'autorange: 'reversed'), constrained to the domain (ie. 'constrain: 'domain'') and it will have the same scale as its x axis (ie. 'scaleanchor: 'x,') in order for pixels to be rendered as squares.

Usage

```
add_image(fig, ...)

## Default S3 method:
add_image(
  fig,
  ...,
  data = NULL,
  colormodel = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
```

```
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  source = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x0 = NULL,
  xaxis = NULL,
  y0 = NULL,
  yaxis = NULL,
  z = NULL,
  zmax = NULL,
  zmin = NULL,
  zorder = NULL,
  zsmooth = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_image(
  arg = NULL,
  ...,
  colormodel = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  source = NULL,
  text = NULL,
```

```
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x0 = NULL,
  xaxis = NULL,
  y0 = NULL,
  yaxis = NULL,
  z = NULL,
  zmax = NULL,
  zmin = NULL,
  zorder = NULL,
  zsmooth = NULL,
  skip_invalid = FALSE
)

go_image_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_image_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_image_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
```

```
)

go_image_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
colormodel	Color model used to map the numerical color components described in ‘z‘ into colors. If ‘source‘ is specified, this attribute will be set to ‘rgba256‘ otherwise it defaults to ‘rgb‘. One of "rgb", "rgba", "rgba256", "hsl", "hsla".
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
dx	Set the pixel's horizontal size. Number.
dy	Set the pixel's vertical size Number.
hoverinfo	Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of "x", "y", "z", "color", "name", "text" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_image_hoverlabel() object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to

those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'z', 'color' and 'colormodel'. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Same as 'text'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgrouptitle`
: Named list or `go_image_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of

	the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.
`name`	Sets the trace name. The trace name appears as the legend item and on hover. String.
`opacity`	Sets the opacity of the trace. Number in the interval [0, 1].
`source`	Specifies the data URI of the image to be visualized. The URI consists of "data:image/[<media subtype\\>][;base64\\],<data\\>" String.
`text`	`go_image()`: Sets the text elements associated with each z value. Data array: a vector, factor, Date/POSIXct vector, matrix or list. `go_image_legendgrouptitle()`: Sets the title of the legend group. String.
`uid`	Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.
`uirevision`	Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.
`visible`	Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.
`x0`	Set the image’s x position. The left edge of the image (or the right edge if the x axis is reversed or dx is negative) will be found at xmin=x0-dx/2 Any value.
`xaxis`	Sets a reference between this trace’s x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...
`y0`	Set the image’s y position. The top edge of the image (or the bottom edge if the y axis is NOT reversed or if dy is negative) will be found at ymin=y0-dy/2. By default when an image trace is included, the y axis will be reversed so that the image is right-side-up, but you can disable this by setting yaxis.autorange=true or by providing an explicit y axis range. Any value.
`yaxis`	Sets a reference between this trace’s y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...
`z`	A 2-dimensional array in which each element is an array of 3 or 4 numbers representing a color. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

zmax — Array defining the higher bound for each color component. Note that the default value will depend on the colormodel. For the 'rgb' colormodel, it is [255, 255, 255]. For the 'rgba' colormodel, it is [255, 255, 255, 1]. For the 'rgba256' colormodel, it is [255, 255, 255, 255]. For the 'hsl' colormodel, it is [360, 100, 100]. For the 'hsla' colormodel, it is [360, 100, 100, 1]. Vector or list of 4 element(s).

zmin — Array defining the lower bound for each color component. Note that the default value will depend on the colormodel. For the 'rgb' colormodel, it is [0, 0, 0]. For the 'rgba' colormodel, it is [0, 0, 0, 0]. For the 'rgba256' colormodel, it is [0, 0, 0, 0]. For the 'hsl' colormodel, it is [0, 0, 0]. For the 'hsla' colormodel, it is [0, 0, 0, 0]. Vector or list of 4 element(s).

zorder — Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher 'zorder' appear in front of those with lower 'zorder'. Integer.

zsmooth — Picks a smoothing algorithm used to smooth 'z' data. This only applies for image traces that use the 'source' attribute. One of `"fast"`, `FALSE`.

row — Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

col — Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

secondary_y — If `TRUE`, use the secondary y axis of the subplot.

skip_invalid — If `TRUE`, silently drop invalid properties instead of raising an error.

arg — Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

align — Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

bgcolor — Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

bordercolor — Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

font — `go_image_hoverlabel()`: Sets the font used in hover labels. Named list or `go_image_hoverlabel_font()` object.

`go_image_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_image_legendgrouptitle_font()` object.

namelength — Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow — Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color
: go_image_hoverlabel_font(): Color: a CSS color string or R color name, or a vector of these.

  go_image_legendgrouptitle_font(): Color: a CSS color string or R color name.

family
: go_image_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

  go_image_legendgrouptitle_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: go_image_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

  go_image_legendgrouptitle_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow
: go_image_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

  go_image_legendgrouptitle_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: go_image_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

  go_image_legendgrouptitle_font(): Number in the interval [1, Inf].

style
: go_image_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

  go_image_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_image_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

  go_image_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_image_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

  go_image_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_image_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

  go_image_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

Details

go_image() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_image() creates an object and adds it to a figure.

Nested objects documented here:

go_image_hoverlabel() image.hoverlabel

go_image_hoverlabel_font() image.hoverlabel.font: Sets the font used in hover labels.

go_image_legendgrouptitle() image.legendgrouptitle

go_image_legendgrouptitle_font() image.legendgrouptitle.font: Sets this legend group's title font.

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_image()
```

---

add_indicator *Indicator traces*

Description

An indicator is used to visualize a single ‘value‘ along with some contextual information such as ‘steps‘ or a ‘threshold‘, using a combination of three visual elements: a number, a delta, and/or a gauge. Deltas are taken with respect to a ‘reference‘. Gauges can be either angular or bullet (aka linear) gauges.

Usage

```
add_indicator(fig, ...)

## Default S3 method:
add_indicator(
  fig,
  ...,
  data = NULL,
  align = NULL,
  customdata = NULL,
  delta = NULL,
  domain = NULL,
  gauge = NULL,
  ids = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  number = NULL,
  title = NULL,
  uid = NULL,
  uirevision = NULL,
  value = NULL,
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_indicator(
  arg = NULL,
  ...,
  align = NULL,
  customdata = NULL,
  delta = NULL,
  domain = NULL,
  gauge = NULL,
  ids = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
```

```
  number = NULL,
  title = NULL,
  uid = NULL,
  uirevision = NULL,
  value = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_indicator_delta(
  arg = NULL,
  ...,
  decreasing = NULL,
  font = NULL,
  increasing = NULL,
  position = NULL,
  prefix = NULL,
  reference = NULL,
  relative = NULL,
  suffix = NULL,
  valueformat = NULL,
  skip_invalid = FALSE
)

go_indicator_delta_decreasing(
  arg = NULL,
  ...,
  color = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_indicator_delta_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_indicator_delta_increasing(
```

```
  arg = NULL,
  ...,
  color = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_indicator_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_indicator_gauge(
  arg = NULL,
  ...,
  axis = NULL,
  bar = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  shape = NULL,
  steps = NULL,
  threshold = NULL,
  skip_invalid = FALSE
)

go_indicator_gauge_axis(
  arg = NULL,
  ...,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  minexponent = NULL,
  nticks = NULL,
  range = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
```

```
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_indicator_gauge_axis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_indicator_gauge_axis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_indicator_gauge_bar(
  arg = NULL,
  ...,
  color = NULL,
  line = NULL,
  thickness = NULL,
```

```
  skip_invalid = FALSE
)

go_indicator_gauge_bar_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_indicator_gauge_step(
  arg = NULL,
  ...,
  color = NULL,
  line = NULL,
  name = NULL,
  range = NULL,
  templateitemname = NULL,
  thickness = NULL,
  skip_invalid = FALSE
)

go_indicator_gauge_step_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_indicator_gauge_threshold(
  arg = NULL,
  ...,
  line = NULL,
  thickness = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_indicator_gauge_threshold_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)
```

```
  go_indicator_legendgrouptitle(
    arg = NULL,
    ...,
    font = NULL,
    text = NULL,
    skip_invalid = FALSE
  )

  go_indicator_legendgrouptitle_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_indicator_number(
    arg = NULL,
    ...,
    font = NULL,
    prefix = NULL,
    suffix = NULL,
    valueformat = NULL,
    skip_invalid = FALSE
  )

  go_indicator_number_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )
```

```
go_indicator_title(
  arg = NULL,
  ...,
  align = NULL,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_indicator_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
align	go_indicator(): Sets the horizontal alignment of the ‘text‘ within the box. Note that this attribute has no effect if an angular gauge is displayed: in this case, it is always centered One of "left", "center", "right". go_indicator_title(): Sets the horizontal alignment of the title. It defaults to ‘center‘ except for bullet charts for which it defaults to right. One of "left", "center", "right".
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
delta	Named list or go_indicator_delta() object.
domain	Named list or go_indicator_domain() object.
gauge	The gauge of the Indicator plot. Named list or go_indicator_gauge() object.

ids — Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

legend — Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: "legend", "legend2", "legend3", ...

legendgrouptitle — Named list or go_indicator_legendgrouptitle() object.

legendrank — Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

legendwidth — Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

meta — Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

mode — Determines how the value is displayed on the graph. ‘number‘ displays the value numerically in text. ‘delta‘ displays the difference to a reference value in text. Finally, ‘gauge‘ displays the value graphically on an axis. Any combination of "number", "delta", "gauge" joined with "+".

name — go_indicator(): Sets the trace name. The trace name appears as the legend item and on hover. String.

go_indicator_gauge_axis_tickformatstop(),go_indicator_gauge_step(): When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

number — Named list or go_indicator_number() object.

title — String (the title text), named list or go_indicator_title() object.

uid — Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision — Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘

	is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(x|y)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.
value	`go_indicator()`: Sets the number to be displayed. Number. `go_indicator_gauge_axis_tickformatstop()`: string - dtickformat for described zoom level, the same as "tickformat" String. `go_indicator_gauge_threshold()`: Sets a treshold value drawn as a line. Number.
visible	`go_indicator()`: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`. `go_indicator_gauge_axis()`: A single toggle to hide the axis while preserving interaction like dragging. Default is true when a cheater plot is present on the axis, otherwise false Logical.
row	`add_indicator()`: Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells. `go_indicator_domain()`: If there is a layout grid, use the domain for this row in the grid for this indicator trace. Integer in the interval [0, Inf].
col	Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.
skip_invalid	If `TRUE`, silently drop invalid properties instead of raising an error.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
decreasing	Named list or `go_indicator_delta_decreasing()` object.
font	`go_indicator_delta()`: Set the font used to display the delta Named list or `go_indicator_delta_font()` object. `go_indicator_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_indicator_legendgrouptitle_font()` object. `go_indicator_number()`: Set the font used to display main number Named list or `go_indicator_number_font()` object. `go_indicator_title()`: Set the font used to display the title Named list or `go_indicator_title_font()` object.
increasing	Named list or `go_indicator_delta_increasing()` object.
position	Sets the position of delta with respect to the number. One of `"top"`, `"bottom"`, `"left"`, `"right"`.
prefix	`go_indicator_delta()`: Sets a prefix appearing before the delta. String. `go_indicator_number()`: Sets a prefix appearing before the number. String.
reference	Sets the reference value to compute the delta. By default, it is set to the current value. Number.

relative	Show relative change Logical.
suffix	go_indicator_delta(): Sets a suffix appearing next to the delta. String. go_indicator_number(): Sets a suffix appearing next to the number. String.
valueformat	Sets the value formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. String.
color	go_indicator_delta_decreasing(), go_indicator_delta_increasing(): Sets the color for increasing value. Color: a CSS color string or R color name. go_indicator_delta_font(), go_indicator_gauge_axis_tickfont(), go_indicator_legendgrou go_indicator_number_font(), go_indicator_title_font(): Color: a CSS color string or R color name. go_indicator_gauge_bar(), go_indicator_gauge_step(): Sets the background color of the arc. Color: a CSS color string or R color name. go_indicator_gauge_bar_line(), go_indicator_gauge_step_line(): Sets the color of the line enclosing each sector. Color: a CSS color string or R color name. go_indicator_gauge_threshold_line(): Sets the color of the threshold line. Color: a CSS color string or R color name.
symbol	Sets the symbol to display for increasing value String.
family	HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.
lineposition	Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".
shadow	Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.
size	Number in the interval [1, Inf].
style	Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".
textcase	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".
variant	Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
weight	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
column	If there is a layout grid, use the domain for this column in the grid for this indicator trace. Integer in the interval [0, Inf].
x	Sets the horizontal domain of this indicator trace (in plot fraction). Vector or list of 2 element(s).
y	Sets the vertical domain of this indicator trace (in plot fraction). Vector or list of 2 element(s).

axis	Named list or go_indicator_gauge_axis() object.
bar	Set the appearance of the gauge's value Named list or go_indicator_gauge_bar() object.
bgcolor	Sets the gauge background color. Color: a CSS color string or R color name.
bordercolor	Sets the color of the border enclosing the gauge. Color: a CSS color string or R color name.
borderwidth	Sets the width (in px) of the border enclosing the gauge. Number in the interval [0, Inf].
shape	Set the shape of the gauge One of "angular", "bullet".
steps	List of named lists or go_indicator_gauge_step() objects.
threshold	Named list or go_indicator_gauge_threshold() object.
dtick	Sets the step in-between ticks on this axis. Use with 'tick0'. Must be a positive number, or special strings available to "log" and "date" axes. If the axis 'type' is "log", then ticks are set every 10^(n*dtick) where n is the tick number. For example, to set a tick mark at 1, 10, 100, 1000, ... set dtick to 1. To set tick marks at 1, 100, 10000, ... set dtick to 2. To set tick marks at 1, 5, 25, 125, 625, 3125, ... set dtick to log_10(5), or 0.69897000433. "log" has several special values; "L<f>", where 'f' is a positive number, gives ticks linearly spaced in value (but not position). For example 'tick0' = 0.1, 'dtick' = "L0.5" will put ticks at 0.1, 0.6, 1.1, 1.6 etc. To show powers of 10 plus small digits between, use "D1" (all digits) or "D2" (only 2 and 5). 'tick0' is ignored for "D1" and "D2". If the axis 'type' is "date", then you must convert the time to milliseconds. For example, to set the interval between ticks to one day, set 'dtick' to 86400000.0. "date" also has special values "M<n>" gives ticks spaced by a number of months. 'n' must be a positive integer. To set ticks on the 15th of every third month, set 'tick0' to "2000-01-15" and 'dtick' to "M3". To set ticks every 4 years, set 'dtick' to "M48" Any value.
exponentformat	Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".
labelalias	Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.
minexponent	Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`range` Sets the range of this axis. Vector or list of 2 element(s).

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`tick0` Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_indicator_gauge_axis_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops` List of named lists or `go_indicator_gauge_axis_tickformatstop()` objects.

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`dtickrange` range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

`enabled` Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

`templateitemname` Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`line` `go_indicator_gauge_bar()`: Named list or `go_indicator_gauge_bar_line()` object.

`go_indicator_gauge_step()`: Named list or `go_indicator_gauge_step_line()` object.

`go_indicator_gauge_threshold()`: Named list or `go_indicator_gauge_threshold_line()` object.

`thickness` `go_indicator_gauge_bar()`, `go_indicator_gauge_step()`: Sets the thickness of the bar as a fraction of the total thickness of the gauge. Number in the interval [0, 1].

`go_indicator_gauge_threshold()`: Sets the thickness of the threshold line as a fraction of the thickness of the gauge. Number in the interval [0, 1].

`width` `go_indicator_gauge_bar_line()`, `go_indicator_gauge_step_line()`: Sets the width (in px) of the line enclosing each sector. Number in the interval [0, Inf].

`go_indicator_gauge_threshold_line()`: Sets the width (in px) of the threshold line. Number in the interval [0, Inf].

`text` `go_indicator_legendgrouptitle()`: Sets the title of the legend group. String.
`go_indicator_title()`: Sets the title of this indicator. String.

Details

`go_indicator()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_indicator()` creates an object and adds it to a figure.

Nested objects documented here:

`go_indicator_delta()` `indicator.delta`

`go_indicator_delta_decreasing()` `indicator.delta.decreasing`

`go_indicator_delta_font()` `indicator.delta.font`: Set the font used to display the delta

`go_indicator_delta_increasing()` `indicator.delta.increasing`

`go_indicator_domain()` `indicator.domain`

`go_indicator_gauge()` `indicator.gauge`: The gauge of the Indicator plot.

`go_indicator_gauge_axis()` `indicator.gauge.axis`

`go_indicator_gauge_axis_tickfont()` `indicator.gauge.axis.tickfont`: Sets the color bar's tick label font

`go_indicator_gauge_axis_tickformatstop()` `indicator.gauge.axis.tickformatstop`

`go_indicator_gauge_bar()` `indicator.gauge.bar`: Set the appearance of the gauge's value

`go_indicator_gauge_bar_line()` `indicator.gauge.bar.line`

`go_indicator_gauge_step()` `indicator.gauge.step`

`go_indicator_gauge_step_line()` `indicator.gauge.step.line`

`go_indicator_gauge_threshold()` `indicator.gauge.threshold`

`go_indicator_gauge_threshold_line()` `indicator.gauge.threshold.line`

`go_indicator_legendgrouptitle()` `indicator.legendgrouptitle`

`go_indicator_legendgrouptitle_font()` `indicator.legendgrouptitle.font`: Sets this legend group's title font.

`go_indicator_number()` `indicator.number`

`go_indicator_number_font()` `indicator.number.font`: Set the font used to display main number

`go_indicator_title()` `indicator.title`

`go_indicator_title_font()` `indicator.title.font`: Set the font used to display the title

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_indicator()
```

---

add_isosurface *Isosurface traces*

Description

Draws isosurfaces between iso-min and iso-max values with coordinates given by four 1-dimensional arrays containing the ‘value‘, ‘x‘, ‘y‘ and ‘z‘ of every vertex of a uniform or non-uniform 3-D grid. Horizontal or vertical slices, caps as well as spaceframe between iso-min and iso-max values could also be drawn using this trace.

Usage

```
add_isosurface(fig, ...)

## Default S3 method:
add_isosurface(
  fig,
  ...,
  data = NULL,
  autocolorscale = NULL,
  caps = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  contour = NULL,
  customdata = NULL,
  flatshading = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  isomax = NULL,
  isomin = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
```

```
  legendwidth = NULL,
  lighting = NULL,
  lightposition = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  reversescale = NULL,
  scene = NULL,
  showlegend = NULL,
  showscale = NULL,
  slices = NULL,
  spaceframe = NULL,
  surface = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  value = NULL,
  valuehoverformat = NULL,
  visible = NULL,
  x = NULL,
  xhoverformat = NULL,
  y = NULL,
  yhoverformat = NULL,
  z = NULL,
  zhoverformat = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_isosurface(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  caps = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  contour = NULL,
  customdata = NULL,
  flatshading = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
```

```
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  isomax = NULL,
  isomin = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lighting = NULL,
  lightposition = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  reversescale = NULL,
  scene = NULL,
  showlegend = NULL,
  showscale = NULL,
  slices = NULL,
  spaceframe = NULL,
  surface = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  value = NULL,
  valuehoverformat = NULL,
  visible = NULL,
  x = NULL,
  xhoverformat = NULL,
  y = NULL,
  yhoverformat = NULL,
  z = NULL,
  zhoverformat = NULL,
  skip_invalid = FALSE
)

go_isosurface_caps(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_isosurface_caps_x(
  arg = NULL,
```

```
  ...,
  fill = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_isosurface_caps_y(
  arg = NULL,
  ...,
  fill = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_isosurface_caps_z(
  arg = NULL,
  ...,
  fill = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_isosurface_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
```

```
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_isosurface_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_isosurface_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
```

```
  value = NULL,
  skip_invalid = FALSE
)

go_isosurface_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_isosurface_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_isosurface_contour(
  arg = NULL,
  ...,
  color = NULL,
  show = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_isosurface_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)
```

```
  go_isosurface_hoverlabel_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_isosurface_legendgrouptitle(
    arg = NULL,
    ...,
    font = NULL,
    text = NULL,
    skip_invalid = FALSE
  )

  go_isosurface_legendgrouptitle_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_isosurface_lighting(
    arg = NULL,
    ...,
    ambient = NULL,
    diffuse = NULL,
    facenormalsepsilon = NULL,
    fresnel = NULL,
    roughness = NULL,
    specular = NULL,
```

```
  vertexnormalsepsilon = NULL,
  skip_invalid = FALSE
)

go_isosurface_lightposition(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_isosurface_slices(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_isosurface_slices_x(
  arg = NULL,
  ...,
  fill = NULL,
  locations = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_isosurface_slices_y(
  arg = NULL,
  ...,
  fill = NULL,
  locations = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_isosurface_slices_z(
  arg = NULL,
  ...,
  fill = NULL,
  locations = NULL,
  show = NULL,
  skip_invalid = FALSE
)
```

```
go_isosurface_spaceframe(
  arg = NULL,
  ...,
  fill = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_isosurface_surface(
  arg = NULL,
  ...,
  count = NULL,
  fill = NULL,
  pattern = NULL,
  show = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
autocolorscale	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
caps	Named list or go_isosurface_caps() object.
cauto	Determines whether or not the color domain is computed with respect to the input data (here ‘value‘) or the bounds set in ‘cmin‘ and ‘cmax‘ Defaults to ‘false‘ when ‘cmin‘ and ‘cmax‘ are set by the user. Logical.
cmax	Sets the upper bound of the color domain. Value should have the same units as ‘value‘ and if set, ‘cmin‘ must be set as well. Number.
cmid	Sets the mid-point of the color domain by scaling ‘cmin‘ and/or ‘cmax‘ to be equidistant to this point. Value should have the same units as ‘value‘. Has no effect when ‘cauto‘ is ‘false‘. Number.
cmin	Sets the lower bound of the color domain. Value should have the same units as ‘value‘ and if set, ‘cmax‘ must be set as well. Number.
coloraxis	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes

are set in the layout, under 'layout.coloraxis', 'layout.coloraxis2', etc. Note that multiple color scales can be linked to the same color axis. Subplot id: "coloraxis", "coloraxis2", "coloraxis3", ...

colorbar — Named list or go_isosurface_colorbar() object.

colorscale — Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the bounds of the colorscale in color space, use 'cmin' and 'cmax'. Alternatively, 'colorscale' may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

contour — Named list or go_isosurface_contour() object.

customdata — Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

flatshading — Determines whether or not normal smoothing is applied to the meshes, creating meshes with an angular, low-poly look via flat reflections. Logical.

hoverinfo — Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of "x", "y", "z", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.

hoverlabel — Named list or go_isosurface_hoverlabel() object.

hovertemplate — Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Same as 'text'. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`isomax`
: Sets the maximum boundary for iso-surface plot. Number.

`isomin`
: Sets the minimum boundary for iso-surface plot. Number.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_isosurface_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`lighting`
: Named list or `go_isosurface_lighting()` object.

`lightposition`
: Named list or `go_isosurface_lightposition()` object.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

`name`
: `go_isosurface()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

  `go_isosurface_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

opacity
: Sets the opacity of the surface. Please note that in the case of using high ‘opacity‘ values for example a value greater than or equal to 0.5 on two surfaces (and 0.25 with four surfaces), an overlay of multiple transparent surfaces may not perfectly be sorted in depth by the webgl API. This behavior may be improved in the near future and is subject to change. Number in the interval [0, 1].

reversescale
: Reverses the color mapping if true. If true, ‘cmin‘ will correspond to the last color in the array and ‘cmax‘ will correspond to the first color. Logical.

scene
: Sets a reference between this trace's 3D coordinate system and a 3D scene. If "scene" (the default value), the (x,y,z) coordinates refer to ‘layout.scene‘. If "scene2", the (x,y,z) coordinates refer to ‘layout.scene2‘, and so on. Subplot id: `"scene"`, `"scene2"`, `"scene3"`, ...

showlegend
: Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

showscale
: Determines whether or not a colorbar is displayed for this trace. Logical.

slices
: Named list or `go_isosurface_slices()` object.

spaceframe
: Named list or `go_isosurface_spaceframe()` object.

surface
: Named list or `go_isosurface_surface()` object.

text
: `go_isosurface()`: Sets the text elements associated with the vertices. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

  `go_isosurface_colorbar_title()`: Sets the title of the color bar. String.

  `go_isosurface_legendgrouptitle()`: Sets the title of the legend group. String.

uid
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision
: Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(x|y)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

value
: `go_isosurface()`: Sets the 4th dimension (value) of the vertices. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  `go_isosurface_colorbar_tickformatstop()`: string - dtickformat for described zoom level, the same as "tickformat" String.

valuehoverformat
: Sets the hover text formatting rule for ‘value‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

visible — Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.

x — go_isosurface(): Sets the X coordinates of the vertices on X axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_isosurface_caps(): Named list or go_isosurface_caps_x() object.

go_isosurface_colorbar(): Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

go_isosurface_lightposition(): Numeric vector, representing the X coordinate for each vertex. Number in the interval [-100000, 100000].

go_isosurface_slices(): Named list or go_isosurface_slices_x() object.

xhoverformat — Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

y — go_isosurface(): Sets the Y coordinates of the vertices on Y axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_isosurface_caps(): Named list or go_isosurface_caps_y() object.

go_isosurface_colorbar(): Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

go_isosurface_lightposition(): Numeric vector, representing the Y coordinate for each vertex. Number in the interval [-100000, 100000].

go_isosurface_slices(): Named list or go_isosurface_slices_y() object.

yhoverformat — Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

z
: go_isosurface(): Sets the Z coordinates of the vertices on Z axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  go_isosurface_caps(): Named list or go_isosurface_caps_z() object.

  go_isosurface_lightposition(): Numeric vector, representing the Z coordinate for each vertex. Number in the interval [-100000, 100000].

  go_isosurface_slices(): Named list or go_isosurface_slices_z() object.

zhoverformat
: Sets the hover text formatting rule for ‘z‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘zaxis.hoverformat‘. String.

row
: Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.

col
: Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

skip_invalid
: If TRUE, silently drop invalid properties instead of raising an error.

arg
: Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

fill
: go_isosurface_caps_x(),go_isosurface_caps_y(),go_isosurface_caps_z(): Sets the fill ratio of the ‘caps‘. The default fill value of the ‘caps‘ is 1 meaning that they are entirely shaded. On the other hand Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Number in the interval [0, 1].

  go_isosurface_slices_x(),go_isosurface_slices_y(),go_isosurface_slices_z(): Sets the fill ratio of the ‘slices‘. The default fill value of the ‘slices‘ is 1 meaning that they are entirely shaded. On the other hand Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Number in the interval [0, 1].

  go_isosurface_spaceframe(): Sets the fill ratio of the ‘spaceframe‘ elements. The default fill value is 0.15 meaning that only 15% of the area of every faces of tetras would be shaded. Applying a greater ‘fill‘ ratio would allow the creation of stronger elements or could be sued to have entirely closed areas (in case of using 1). Number in the interval [0, 1].

  go_isosurface_surface(): Sets the fill ratio of the iso-surface. The default fill value of the surface is 1 meaning that they are entirely shaded. On the other hand Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Number in the interval [0, 1].

show
: go_isosurface_caps_x(): Sets the fill ratio of the ‘slices‘. The default fill value of the x ‘slices‘ is 1 meaning that they are entirely shaded. On the other hand Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Logical.

go_isosurface_caps_y(): Sets the fill ratio of the ‘slices‘. The default fill value of the y ‘slices‘ is 1 meaning that they are entirely shaded. On the other hand Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Logical.

go_isosurface_caps_z(): Sets the fill ratio of the ‘slices‘. The default fill value of the z ‘slices‘ is 1 meaning that they are entirely shaded. On the other hand Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Logical.

go_isosurface_contour(): Sets whether or not dynamic contours are shown on hover Logical.

go_isosurface_slices_x(): Determines whether or not slice planes about the x dimension are drawn. Logical.

go_isosurface_slices_y(): Determines whether or not slice planes about the y dimension are drawn. Logical.

go_isosurface_slices_z(): Determines whether or not slice planes about the z dimension are drawn. Logical.

go_isosurface_spaceframe(): Displays/hides tetrahedron shapes between minimum and maximum iso-values. Often useful when either caps or surfaces are disabled or filled with values less than 1. Logical.

go_isosurface_surface(): Hides/displays surfaces between minimum and maximum iso-values. Logical.

`bgcolor` go_isosurface_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name.

go_isosurface_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`bordercolor` go_isosurface_colorbar(): Sets the axis line color. Color: a CSS color string or R color name.

go_isosurface_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels

we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar's tick label font Named list or `go_isosurface_colorbar_tickfont()` object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or `go_isosurface_colorbar_tickformatstop()` objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

ticklabelposition Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

ticklabelstep Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

ticklen Sets the tick length (in px). Number in the interval [0, Inf].

tickmode Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

tickprefix Sets a tick label prefix. String.

ticks Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

ticksuffix Sets a tick label suffix. String.

ticktext Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals — Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth — Sets the tick width (in px). Number in the interval [0, Inf].

title — String (the title text), named list or go_isosurface_colorbar_title() object.

xanchor — Sets this color bar's horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of "left", "center", "right".

xpad — Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".

yanchor — Sets this color bar's vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

color — go_isosurface_colorbar_tickfont(), go_isosurface_colorbar_title_font(), go_isosurface_legendgrouptitle_font(): Color: a CSS color string or R color name.

go_isosurface_contour(): Sets the color of the contour lines. Color: a CSS color string or R color name.

go_isosurface_hoverlabel_font(): Color: a CSS color string or R color name, or a vector of these.

family — go_isosurface_colorbar_tickfont(), go_isosurface_colorbar_title_font(), go_isosurface_legendgrouptitle_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

go_isosurface_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

lineposition — go_isosurface_colorbar_tickfont(), go_isosurface_colorbar_title_font(), go_isosurface_legendgrouptitle_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g.

"under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

go_isosurface_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

shadow

go_isosurface_colorbar_tickfont(), go_isosurface_colorbar_title_font(), go_isosurface_legendgrouptitle_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

go_isosurface_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size

go_isosurface_colorbar_tickfont(), go_isosurface_colorbar_title_font(), go_isosurface_legendgrouptitle_font(): Number in the interval [1, Inf].

go_isosurface_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

style

go_isosurface_colorbar_tickfont(), go_isosurface_colorbar_title_font(), go_isosurface_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

go_isosurface_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

textcase

go_isosurface_colorbar_tickfont(), go_isosurface_colorbar_title_font(), go_isosurface_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

go_isosurface_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant

go_isosurface_colorbar_tickfont(), go_isosurface_colorbar_title_font(), go_isosurface_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

go_isosurface_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight

go_isosurface_colorbar_tickfont(), go_isosurface_colorbar_title_font(), go_isosurface_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

go_isosurface_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

`dtickrange` range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

`enabled` Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

`templateitemname` Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`font` `go_isosurface_colorbar_title()`: Sets this color bar's title font. Named list or `go_isosurface_colorbar_title_font()` object.

`go_isosurface_hoverlabel()`: Sets the font used in hover labels. Named list or `go_isosurface_hoverlabel_font()` object.

`go_isosurface_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_isosurface_legendgrouptitle_font()` object.

`side` Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.

`width` Sets the width of the contour lines. Number in the interval [1, 16].

`align` Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`namelength` Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to ‘namelength - 3‘ characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

`showarrow` Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

`ambient` Ambient light increases overall color visibility but can wash out the image. Number in the interval [0, 1].

`diffuse` Represents the extent that incident rays are reflected in a range of angles. Number in the interval [0, 1].

`facenormalsepsilon` Epsilon for face normals calculation avoids math issues arising from degenerate geometry. Number in the interval [0, 1].

`fresnel` Represents the reflectance as a dependency of the viewing angle; e.g. paper is reflective when viewing it from the edge of the paper (almost 90 degrees), causing shine. Number in the interval [0, 5].

`roughness` Alters specular reflection; the rougher the surface, the wider and less contrasty the shine. Number in the interval [0, 1].

specular
: Represents the level that incident rays are reflected in a single direction, causing shine. Number in the interval [0, 2].

vertexnormalsepsilon
: Epsilon for vertex normals calculation avoids math issues arising from degenerate geometry. Number in the interval [0, 1].

locations
: go_isosurface_slices_x(): Specifies the location(s) of slices on the axis. When not specified slices would be created for all points of the axis x except start and end. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  go_isosurface_slices_y(): Specifies the location(s) of slices on the axis. When not specified slices would be created for all points of the axis y except start and end. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  go_isosurface_slices_z(): Specifies the location(s) of slices on the axis. When not specified slices would be created for all points of the axis z except start and end. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

count
: Sets the number of iso-surfaces between minimum and maximum iso-values. By default this value is 2 meaning that only minimum and maximum surfaces would be drawn. Integer in the interval [1, Inf].

pattern
: Sets the surface pattern of the iso-surface 3-D sections. The default pattern of the surface is 'all' meaning that the rest of surface elements would be shaded. The check options (either 1 or 2) could be used to draw half of the squares on the surface. Using various combinations of capital 'A', 'B', 'C', 'D' and 'E' may also be used to reduce the number of triangles on the iso-surfaces and creating other patterns of interest. Any combination of "A", "B", "C", "D", "E" joined with "+", or one of "all", "odd", "even".

Details

go_isosurface() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_isosurface() creates an object and adds it to a figure.

Nested objects documented here:

go_isosurface_caps() isosurface.caps

go_isosurface_caps_x() isosurface.caps.x

go_isosurface_caps_y() isosurface.caps.y

go_isosurface_caps_z() isosurface.caps.z

go_isosurface_colorbar() isosurface.colorbar

go_isosurface_colorbar_tickfont() isosurface.colorbar.tickfont: Sets the color bar's tick label font

go_isosurface_colorbar_tickformatstop() isosurface.colorbar.tickformatstop

go_isosurface_colorbar_title() isosurface.colorbar.title

go_isosurface_colorbar_title_font() isosurface.colorbar.title.font: Sets this color bar's title font.

go_isosurface_contour() isosurface.contour

`go_isosurface_hoverlabel()` `isosurface.hoverlabel`

`go_isosurface_hoverlabel_font()` `isosurface.hoverlabel.font`: Sets the font used in hover labels.

`go_isosurface_legendgrouptitle()` `isosurface.legendgrouptitle`

`go_isosurface_legendgrouptitle_font()` `isosurface.legendgrouptitle.font`: Sets this legend group's title font.

`go_isosurface_lighting()` `isosurface.lighting`

`go_isosurface_lightposition()` `isosurface.lightposition`

`go_isosurface_slices()` `isosurface.slices`

`go_isosurface_slices_x()` `isosurface.slices.x`

`go_isosurface_slices_y()` `isosurface.slices.y`

`go_isosurface_slices_z()` `isosurface.slices.z`

`go_isosurface_spaceframe()` `isosurface.spaceframe`

`go_isosurface_surface()` `isosurface.surface`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_isosurface(x = 1:3, y = c(2, 1, 3))
```

---

add_layout_image *Layout image objects*

Description

Properties of `layout.image`.

Usage

```
add_layout_image(
  fig,
  arg = NULL,
  ...,
  layer = NULL,
  name = NULL,
  opacity = NULL,
```

```
  sizex = NULL,
  sizey = NULL,
  sizing = NULL,
  source = NULL,
  templateitemname = NULL,
  visible = NULL,
  x = NULL,
  xanchor = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  yref = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  exclude_empty_subplots = FALSE,
  skip_invalid = FALSE
)

go_layout_image(
  arg = NULL,
  ...,
  layer = NULL,
  name = NULL,
  opacity = NULL,
  sizex = NULL,
  sizey = NULL,
  sizing = NULL,
  source = NULL,
  templateitemname = NULL,
  visible = NULL,
  x = NULL,
  xanchor = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  yref = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.

`layer` Specifies whether images are drawn below or above traces. When ‘xref‘ and ‘yref‘ are both set to ‘paper‘, image is drawn below the entire plot area. One of `"below"`, `"above"`.

`name` When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` Sets the opacity of the image. Number in the interval [0, 1].

`sizex` Sets the image container size horizontally. The image will be sized based on the ‘position‘ value. When ‘xref‘ is set to ‘paper‘, units are sized relative to the plot width. When ‘xref‘ ends with ‘ domain‘, units are sized relative to the axis width. Number.

`sizey` Sets the image container size vertically. The image will be sized based on the ‘position‘ value. When ‘yref‘ is set to ‘paper‘, units are sized relative to the plot height. When ‘yref‘ ends with ‘ domain‘, units are sized relative to the axis height. Number.

`sizing` Specifies which dimension of the image to constrain. One of `"fill"`, `"contain"`, `"stretch"`.

`source` Specifies the URL of the image to be used. The URL must be accessible from the domain where the plot code is run, and can be either relative or absolute. String.

`templateitemname` Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`visible` Determines whether or not this image is visible. Logical.

`x` Sets the image’s x position. When ‘xref‘ is set to ‘paper‘, units are sized relative to the plot height. See ‘xref‘ for more info Any value.

`xanchor` Sets the anchor for the x position One of `"left"`, `"center"`, `"right"`.

`xref` Sets the images’s x coordinate axis. If set to a x axis id (e.g. "x" or "x2"), the ‘x‘ position refers to a x coordinate. If set to "paper", the ‘x‘ position refers to the distance from the left of the plotting area in normalized coordinates where 0 (1) corresponds to the left (right). If set to a x axis ID followed by "domain" (separated by a space), the position behaves like for "paper", but refers to the distance in fractions of the domain length from the left of the domain of that axis: e.g., "x2 domain" refers to the domain of the second x axis and a x position of 0.5 refers to the point between the left and the right of the domain of the second x axis. One of `"paper"`, or a string matching `"^x([2-9]|[1-9][0-9]+)?(domain)?$"`.

`y` Sets the image’s y position. When ‘yref‘ is set to ‘paper‘, units are sized relative to the plot height. See ‘yref‘ for more info Any value.

yanchor	Sets the anchor for the y position. One of "top", "middle", "bottom".
yref	Sets the images's y coordinate axis. If set to a y axis id (e.g. "y" or "y2"), the 'y' position refers to a y coordinate. If set to "paper", the 'y' position refers to the distance from the bottom of the plotting area in normalized coordinates where 0 (1) corresponds to the bottom (top). If set to a y axis ID followed by "domain" (separated by a space), the position behaves like for "paper", but refers to the distance in fractions of the domain length from the bottom of the domain of that axis: e.g., "y2 domain" refers to the domain of the second y axis and a y position of 0.5 refers to the point between the bottom and the top of the domain of the second y axis. One of "paper", or a string matching "^y([2-9]|[1-9][0-9]+)?(domain)?$".
row	Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
secondary_y	If TRUE, use the secondary y axis of the subplot.
exclude_empty_subplots	If TRUE, skip subplots that have no traces or layout objects.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.

Details

go_layout_image() create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_layout_image() creates an object and adds it to a figure.

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

---

add_mesh3d *Mesh3d traces*

Description

Draws sets of triangles with coordinates given by three 1-dimensional arrays in 'x', 'y', 'z' and (1) a sets of 'i', 'j', 'k' indices (2) Delaunay triangulation or (3) the Alpha-shape algorithm or (4) the Convex-hull algorithm

Usage

```
add_mesh3d(fig, ...)

## Default S3 method:
add_mesh3d(
  fig,
  ...,
  data = NULL,
  alphahull = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  contour = NULL,
  customdata = NULL,
  delaunayaxis = NULL,
  facecolor = NULL,
  flatshading = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  i = NULL,
  ids = NULL,
  intensity = NULL,
  intensitymode = NULL,
  j = NULL,
  k = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lighting = NULL,
  lightposition = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  reversescale = NULL,
  scene = NULL,
  showlegend = NULL,
  showscale = NULL,
```

```
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  vertexcolor = NULL,
  visible = NULL,
  x = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  y = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  z = NULL,
  zcalendar = NULL,
  zhoverformat = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_mesh3d(
  arg = NULL,
  ...,
  alphahull = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  contour = NULL,
  customdata = NULL,
  delaunayaxis = NULL,
  facecolor = NULL,
  flatshading = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  i = NULL,
  ids = NULL,
  intensity = NULL,
  intensitymode = NULL,
  j = NULL,
  k = NULL,
```

```
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lighting = NULL,
  lightposition = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  reversescale = NULL,
  scene = NULL,
  showlegend = NULL,
  showscale = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  vertexcolor = NULL,
  visible = NULL,
  x = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  y = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  z = NULL,
  zcalendar = NULL,
  zhoverformat = NULL,
  skip_invalid = FALSE
)

go_mesh3d_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
```

```
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_mesh3d_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
```

```
)

go_mesh3d_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_mesh3d_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_mesh3d_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_mesh3d_contour(
  arg = NULL,
  ...,
  color = NULL,
  show = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_mesh3d_hoverlabel(
  arg = NULL,
```

```
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_mesh3d_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_mesh3d_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_mesh3d_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

```
go_mesh3d_lighting(
  arg = NULL,
  ...,
  ambient = NULL,
  diffuse = NULL,
  facenormalsepsilon = NULL,
  fresnel = NULL,
  roughness = NULL,
  specular = NULL,
  vertexnormalsepsilon = NULL,
  skip_invalid = FALSE
)

go_mesh3d_lightposition(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
alphahull	Determines how the mesh surface triangles are derived from the set of vertices (points) represented by the 'x', 'y' and 'z' arrays, if the 'i', 'j', 'k' arrays are not supplied. For general use of 'mesh3d' it is preferred that 'i', 'j', 'k' are supplied. If "-1", Delaunay triangulation is used, which is mainly suitable if the mesh is a single, more or less layer surface that is perpendicular to 'delaunayaxis'. In case the 'delaunayaxis' intersects the mesh surface at more than one point it will result triangles that are very long in the dimension of 'delaunayaxis'. If ">0", the alpha-shape algorithm is used. In this case, the positive 'alphahull' value signals the use of the alpha-shape algorithm, _and_ its value acts as the parameter for the mesh fitting. If 0, the convex-hull algorithm is used. It is suitable for convex bodies or if the intention is to enclose the 'x', 'y' and 'z' point set into a convex hull. Number.
autocolorscale	Determines whether the colorscale is a default palette ('autocolorscale: true') or the palette determined by 'colorscale'. In case 'colorscale' is unspecified or 'autocolorscale' is true, the default palette will be chosen according to whether numbers in the 'color' array are all positive, all negative or mixed. Logical.

cauto — Determines whether or not the color domain is computed with respect to the input data (here ‘intensity‘) or the bounds set in ‘cmin‘ and ‘cmax‘ Defaults to ‘false‘ when ‘cmin‘ and ‘cmax‘ are set by the user. Logical.

cmax — Sets the upper bound of the color domain. Value should have the same units as ‘intensity‘ and if set, ‘cmin‘ must be set as well. Number.

cmid — Sets the mid-point of the color domain by scaling ‘cmin‘ and/or ‘cmax‘ to be equidistant to this point. Value should have the same units as ‘intensity‘. Has no effect when ‘cauto‘ is ‘false‘. Number.

cmin — Sets the lower bound of the color domain. Value should have the same units as ‘intensity‘ and if set, ‘cmax‘ must be set as well. Number.

color — `go_mesh3d()`: Sets the color of the whole mesh Color: a CSS color string or R color name, or a number mapped to a color by `mesh3d.colorscale`.

`go_mesh3d_colorbar_tickfont()`, `go_mesh3d_colorbar_title_font()`, `go_mesh3d_legendgroupt` Color: a CSS color string or R color name.

`go_mesh3d_contour()`: Sets the color of the contour lines. Color: a CSS color string or R color name.

`go_mesh3d_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

coloraxis — Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

colorbar — Named list or `go_mesh3d_colorbar()` object.

colorscale — Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘cmin‘ and ‘cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

contour — Named list or `go_mesh3d_contour()` object.

customdata — Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

delaunayaxis — Sets the Delaunay axis, which is the axis that is perpendicular to the surface of the Delaunay triangulation. It has an effect if ‘i‘, ‘j‘, ‘k‘ are not provided and ‘alphahull‘ is set to indicate Delaunay triangulation. One of `"x"`, `"y"`, `"z"`.

facecolor — Sets the color of each face Overrides "color" and "vertexcolor". Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`flatshading` Determines whether or not normal smoothing is applied to the meshes, creating meshes with an angular, low-poly look via flat reflections. Logical.

`hoverinfo` Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_mesh3d_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Same as 'text'. String, or a vector of these.

`i` A vector of vertex indices, i.e. integer values between 0 and the length of the vertex vectors, representing the "first" vertex of a triangle. For example, '{i[m], j[m], k[m]}' together represent face m (triangle m) in the mesh, where 'i[m] = n' points to the triplet '{x[n], y[n], z[n]}' in the vertex arrays. Therefore, each element in 'i' represents a point in space, which is the first vertex of a triangle. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`intensity` Sets the intensity values for vertices or cells as defined by 'intensitymode'. It can be used for plotting fields on meshes. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`intensitymode` Determines the source of 'intensity' values. One of `"vertex"`, `"cell"`.

j — A vector of vertex indices, i.e. integer values between 0 and the length of the vertex vectors, representing the "second" vertex of a triangle. For example, ‘{i[m], j[m], k[m]}‘ together represent face m (triangle m) in the mesh, where ‘j[m] = n‘ points to the triplet ‘{x[n], y[n], z[n]}‘ in the vertex arrays. Therefore, each element in ‘j‘ represents a point in space, which is the second vertex of a triangle. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

k — A vector of vertex indices, i.e. integer values between 0 and the length of the vertex vectors, representing the "third" vertex of a triangle. For example, ‘{i[m], j[m], k[m]}‘ together represent face m (triangle m) in the mesh, where ‘k[m] = n‘ points to the triplet ‘{x[n], y[n], z[n]}‘ in the vertex arrays. Therefore, each element in ‘k‘ represents a point in space, which is the third vertex of a triangle. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

legend — Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

legendgroup — Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

legendgrouptitle — Named list or `go_mesh3d_legendgrouptitle()` object.

legendrank — Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

legendwidth — Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

lighting — Named list or `go_mesh3d_lighting()` object.

lightposition — Named list or `go_mesh3d_lightposition()` object.

meta — Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

name — `go_mesh3d()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_mesh3d_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your

modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` Sets the opacity of the surface. Please note that in the case of using high ‘opacity‘ values for example a value greater than or equal to 0.5 on two surfaces (and 0.25 with four surfaces), an overlay of multiple transparent surfaces may not perfectly be sorted in depth by the webgl API. This behavior may be improved in the near future and is subject to change. Number in the interval [0, 1].

`reversescale` Reverses the color mapping if true. If true, ‘cmin‘ will correspond to the last color in the array and ‘cmax‘ will correspond to the first color. Logical.

`scene` Sets a reference between this trace's 3D coordinate system and a 3D scene. If "scene" (the default value), the (x,y,z) coordinates refer to ‘layout.scene‘. If "scene2", the (x,y,z) coordinates refer to ‘layout.scene2‘, and so on. Subplot id: `"scene"`, `"scene2"`, `"scene3"`, ...

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Logical.

`text` `go_mesh3d()`: Sets the text elements associated with the vertices. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_mesh3d_colorbar_title()`: Sets the title of the color bar. String.

`go_mesh3d_legendgrouptitle()`: Sets the title of the legend group. String.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`vertexcolor` Sets the color of each vertex Overrides "color". While Red, green and blue colors are in the range of 0 and 255; in the case of having vertex color data in RGBA format, the alpha color should be normalized to be between 0 and 1. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`visible` Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`x` `go_mesh3d()`: Sets the X coordinates of the vertices. The nth element of vectors ‘x‘, ‘y‘ and ‘z‘ jointly represent the X, Y and Z coordinates of the nth vertex. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_mesh3d_colorbar(): Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

go_mesh3d_lightposition(): Numeric vector, representing the X coordinate for each vertex. Number in the interval [-100000, 100000].

xcalendar — Sets the calendar system to use with ‘x‘ date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

xhoverformat — Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

y — go_mesh3d(): Sets the Y coordinates of the vertices. The nth element of vectors ‘x‘, ‘y‘ and ‘z‘ jointly represent the X, Y and Z coordinates of the nth vertex. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_mesh3d_colorbar(): Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

go_mesh3d_lightposition(): Numeric vector, representing the Y coordinate for each vertex. Number in the interval [-100000, 100000].

ycalendar — Sets the calendar system to use with ‘y‘ date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

yhoverformat — Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

z — go_mesh3d(): Sets the Z coordinates of the vertices. The nth element of vectors ‘x‘, ‘y‘ and ‘z‘ jointly represent the X, Y and Z coordinates of the nth vertex. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_mesh3d_lightposition(): Numeric vector, representing the Z coordinate for each vertex. Number in the interval [-100000, 100000].

zcalendar Sets the calendar system to use with ‘z‘ date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

zhoverformat Sets the hover text formatting rule for ‘z‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘zaxis.hoverformat‘. String.

row Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.

col Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

skip_invalid If TRUE, silently drop invalid properties instead of raising an error.

arg Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

bgcolor go_mesh3d_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name.

go_mesh3d_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

bordercolor go_mesh3d_colorbar(): Sets the axis line color. Color: a CSS color string or R color name.

go_mesh3d_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

borderwidth Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

dtick Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

exponentformat Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

labelalias Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is

wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_mesh3d_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops` List of named lists or `go_mesh3d_colorbar_tickformatstop()` objects.

`ticklabeloverflow` Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition` Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals — Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth — Sets the tick width (in px). Number in the interval [0, Inf].

title — String (the title text), named list or go_mesh3d_colorbar_title() object.

xanchor — Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of "left", "center", "right".

xpad — Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".

yanchor — Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

family — go_mesh3d_colorbar_tickfont(), go_mesh3d_colorbar_title_font(), go_mesh3d_legendgroupt
HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String.

go_mesh3d_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String, or a vector of these.

lineposition — go_mesh3d_colorbar_tickfont(), go_mesh3d_colorbar_title_font(), go_mesh3d_legendgroupt
Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

go_mesh3d_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

shadow — go_mesh3d_colorbar_tickfont(), go_mesh3d_colorbar_title_font(), go_mesh3d_legendgroupt
Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

go_mesh3d_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size
go_mesh3d_colorbar_tickfont(), go_mesh3d_colorbar_title_font(), go_mesh3d_legendgroupt
Number in the interval [1, Inf].

go_mesh3d_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

style
go_mesh3d_colorbar_tickfont(), go_mesh3d_colorbar_title_font(), go_mesh3d_legendgroupt
Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

go_mesh3d_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

textcase
go_mesh3d_colorbar_tickfont(), go_mesh3d_colorbar_title_font(), go_mesh3d_legendgroupt
Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

go_mesh3d_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant
go_mesh3d_colorbar_tickfont(), go_mesh3d_colorbar_title_font(), go_mesh3d_legendgroupt
Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

go_mesh3d_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight
go_mesh3d_colorbar_tickfont(), go_mesh3d_colorbar_title_font(), go_mesh3d_legendgroupt
Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

go_mesh3d_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange
range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled
Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname
Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value
string - dtickformat for described zoom level, the same as "tickformat" String.

font
: go_mesh3d_colorbar_title(): Sets this color bar's title font. Named list or go_mesh3d_colorbar_title_font() object.
go_mesh3d_hoverlabel(): Sets the font used in hover labels. Named list or go_mesh3d_hoverlabel_font() object.
go_mesh3d_legendgrouptitle(): Sets this legend group's title font. Named list or go_mesh3d_legendgrouptitle_font() object.

side
: Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of "right", "top", "bottom".

show
: Sets whether or not dynamic contours are shown on hover Logical.

width
: Sets the width of the contour lines. Number in the interval [1, 16].

align
: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

ambient
: Ambient light increases overall color visibility but can wash out the image. Number in the interval [0, 1].

diffuse
: Represents the extent that incident rays are reflected in a range of angles. Number in the interval [0, 1].

facenormalsepsilon
: Epsilon for face normals calculation avoids math issues arising from degenerate geometry. Number in the interval [0, 1].

fresnel
: Represents the reflectance as a dependency of the viewing angle; e.g. paper is reflective when viewing it from the edge of the paper (almost 90 degrees), causing shine. Number in the interval [0, 5].

roughness
: Alters specular reflection; the rougher the surface, the wider and less contrasty the shine. Number in the interval [0, 1].

specular
: Represents the level that incident rays are reflected in a single direction, causing shine. Number in the interval [0, 2].

vertexnormalsepsilon
: Epsilon for vertex normals calculation avoids math issues arising from degenerate geometry. Number in the interval [0, 1].

Details

go_mesh3d() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_mesh3d() creates an object and adds it to a figure.

Nested objects documented here:

`go_mesh3d_colorbar()` `mesh3d.colorbar`

`go_mesh3d_colorbar_tickfont()` `mesh3d.colorbar.tickfont`: Sets the color bar's tick label font

`go_mesh3d_colorbar_tickformatstop()` `mesh3d.colorbar.tickformatstop`

`go_mesh3d_colorbar_title()` `mesh3d.colorbar.title`

`go_mesh3d_colorbar_title_font()` `mesh3d.colorbar.title.font`: Sets this color bar's title font.

`go_mesh3d_contour()` `mesh3d.contour`

`go_mesh3d_hoverlabel()` `mesh3d.hoverlabel`

`go_mesh3d_hoverlabel_font()` `mesh3d.hoverlabel.font`: Sets the font used in hover labels.

`go_mesh3d_legendgrouptitle()` `mesh3d.legendgrouptitle`

`go_mesh3d_legendgrouptitle_font()` `mesh3d.legendgrouptitle.font`: Sets this legend group's title font.

`go_mesh3d_lighting()` `mesh3d.lighting`

`go_mesh3d_lightposition()` `mesh3d.lightposition`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_mesh3d(x = 1:3, y = c(2, 1, 3))
```

---

add_ohlc *Ohlc traces*

Description

The ohlc (short for Open-High-Low-Close) is a style of financial chart describing open, high, low and close for a given 'x' coordinate (most likely time). The tip of the lines represent the 'low' and 'high' values and the horizontal segments represent the 'open' and 'close' values. Sample points where the close value is higher (lower) than the open value are called increasing (decreasing). By default, increasing items are drawn in green whereas decreasing are drawn in red.

Usage

```
add_ohlc(fig, ...)

## Default S3 method:
add_ohlc(
  fig,
  ...,
  data = NULL,
  close = NULL,
  customdata = NULL,
  decreasing = NULL,
  high = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  increasing = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  low = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  open = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  tickwidth = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  xaxis = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  zorder = NULL,
  row = NULL,
```

```
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_ohlc(
  arg = NULL,
  ...,
  close = NULL,
  customdata = NULL,
  decreasing = NULL,
  high = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  increasing = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  low = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  open = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  tickwidth = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  xaxis = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  zorder = NULL,
  skip_invalid = FALSE
```

```
)

go_ohlc_decreasing(arg = NULL, ..., line = NULL, skip_invalid = FALSE)

go_ohlc_decreasing_line(
  arg = NULL,
  ...,
  color = NULL,
  dash = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_ohlc_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  split = NULL,
  skip_invalid = FALSE
)

go_ohlc_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_ohlc_increasing(arg = NULL, ..., line = NULL, skip_invalid = FALSE)

go_ohlc_increasing_line(
  arg = NULL,
  ...,
  color = NULL,
  dash = NULL,
```

```
  width = NULL,
  skip_invalid = FALSE
)

go_ohlc_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_ohlc_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_ohlc_line(arg = NULL, ..., dash = NULL, width = NULL, skip_invalid = FALSE)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
close	Sets the close values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
decreasing	Named list or go_ohlc_decreasing() object.
high	Sets the high values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`hoverinfo` Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_ohlc_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'open', 'high', 'low' and 'close'. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Same as 'text'. String, or a vector of these.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`increasing` Named list or `go_ohlc_increasing()` object.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle` Named list or `go_ohlc_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line` `go_ohlc()`: Named list or `go_ohlc_line()` object.

`go_ohlc_decreasing()`: Named list or `go_ohlc_decreasing_line()` object.

`go_ohlc_increasing()`: Named list or `go_ohlc_increasing_line()` object.

`low` Sets the low values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and color-bar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` Sets the trace name. The trace name appears as the legend item and on hover. String.

`opacity` Sets the opacity of the trace. Number in the interval [0, 1].

`open` Sets the open values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`text` `go_ohlc()`: Sets hover text elements associated with each sample point. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to this trace’s sample points. String, or a vector of these.

`go_ohlc_legendgrouptitle()`: Sets the title of the legend group. String.

`tickwidth` Sets the width of the open/close tick marks relative to the "x" minimal interval. Number in the interval [0, 0.5].

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision — Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

visible — Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

x — Sets the x coordinates. If absent, linear coordinate will be generated. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

xaxis — Sets a reference between this trace’s x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

xcalendar — Sets the calendar system to use with ‘x‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

xhoverformat — Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

xperiod — Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the x axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

xperiod0 — Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the x0 axis. When ‘x0period‘ is round number of weeks, the ‘x0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

xperiodalignment — Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the x axis. One of `"start"`, `"middle"`, `"end"`.

yaxis — Sets a reference between this trace’s y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...

yhoverformat — Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

zorder — Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher ‘zorder‘ appear in front of those with lower ‘zorder‘. Integer.

row — Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

col — Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

secondary_y — If `TRUE`, use the secondary y axis of the subplot.

skip_invalid — If `TRUE`, silently drop invalid properties instead of raising an error.

arg — Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

color — `go_ohlc_decreasing_line()`, `go_ohlc_increasing_line()`: Sets the line color. Color: a CSS color string or R color name.

`go_ohlc_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

`go_ohlc_legendgrouptitle_font()`: Color: a CSS color string or R color name.

dash — `go_ohlc_decreasing_line()`, `go_ohlc_increasing_line()`: Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

`go_ohlc_line()`: Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). Note that this style setting can also be set per direction via ‘increasing.line.dash‘ and ‘decreasing.line.dash‘. One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

width — `go_ohlc_decreasing_line()`, `go_ohlc_increasing_line()`: Sets the line width (in px). Number in the interval [0, Inf].

`go_ohlc_line()`: Sets the line width (in px). Note that this style setting can also be set per direction via ‘increasing.line.width‘ and ‘decreasing.line.width‘. Number in the interval [0, Inf].

align — Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`bgcolor` Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`bordercolor` Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`font` `go_ohlc_hoverlabel()`: Sets the font used in hover labels. Named list or `go_ohlc_hoverlabel_font()` object.

`go_ohlc_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_ohlc_legendgrouptitle_font()` object.

`namelength` Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

`showarrow` Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

`split` Show hover information (open, close, high, low) in separate labels, rather than a single unified label. Default: FALSE. When set to TRUE, 'hovertemplate' is ignored. Logical.

`family` `go_ohlc_hoverlabel_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

`go_ohlc_legendgrouptitle_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

`lineposition` `go_ohlc_hoverlabel_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

`go_ohlc_legendgrouptitle_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

`shadow` `go_ohlc_hoverlabel_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

`go_ohlc_legendgrouptitle_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: go_ohlc_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

  go_ohlc_legendgrouptitle_font(): Number in the interval [1, Inf].

style
: go_ohlc_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

  go_ohlc_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_ohlc_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

  go_ohlc_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_ohlc_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

  go_ohlc_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_ohlc_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

  go_ohlc_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

Details

go_ohlc() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_ohlc() creates an object and adds it to a figure.

Nested objects documented here:

go_ohlc_decreasing() ohlc.decreasing

go_ohlc_decreasing_line() ohlc.decreasing.line

go_ohlc_hoverlabel() ohlc.hoverlabel

go_ohlc_hoverlabel_font() ohlc.hoverlabel.font: Sets the font used in hover labels.

go_ohlc_increasing() ohlc.increasing

go_ohlc_increasing_line() ohlc.increasing.line

go_ohlc_legendgrouptitle() ohlc.legendgrouptitle

go_ohlc_legendgrouptitle_font() ohlc.legendgrouptitle.font: Sets this legend group's title font.

go_ohlc_line() ohlc.line

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_ohlc()
```

---

add_parcats *Parcats traces*

Description

Parallel categories diagram for multidimensional categorical data.

Usage

```
add_parcats(fig, ...)

## Default S3 method:
add_parcats(
  fig,
  ...,
  data = NULL,
  arrangement = NULL,
  bundlecolors = NULL,
  counts = NULL,
  dimensions = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  labelfont = NULL,
  legendgrouptitle = NULL,
  legendwidth = NULL,
  line = NULL,
  meta = NULL,
  name = NULL,
  sortpaths = NULL,
  tickfont = NULL,
  uid = NULL,
  uirevision = NULL,
```

```
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_parcats(
  arg = NULL,
  ...,
  arrangement = NULL,
  bundlecolors = NULL,
  counts = NULL,
  dimensions = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  labelfont = NULL,
  legendgrouptitle = NULL,
  legendwidth = NULL,
  line = NULL,
  meta = NULL,
  name = NULL,
  sortpaths = NULL,
  tickfont = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_parcats_dimension(
  arg = NULL,
  ...,
  categoryarray = NULL,
  categoryorder = NULL,
  displayindex = NULL,
  label = NULL,
  ticktext = NULL,
  values = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_parcats_domain(
  arg = NULL,
  ...,
```

```
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_parcats_labelfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_parcats_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_parcats_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_parcats_line(
  arg = NULL,
  ...,
```

```
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  reversescale = NULL,
  shape = NULL,
  showscale = NULL,
  skip_invalid = FALSE
)

go_parcats_line_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
```

```
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_parcats_line_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_parcats_line_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_parcats_line_colorbar_title(
```

```
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_parcats_line_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_parcats_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).

`arrangement` Sets the drag interaction mode for categories and dimensions. If 'perpendicular', the categories can only move along a line perpendicular to the paths. If 'freeform', the categories can freely move on the plane. If 'fixed', the categories and dimensions are stationary. One of `"perpendicular"`, `"freeform"`, `"fixed"`.

`bundlecolors` Sort paths so that like colors are bundled together within each category. Logical.

`counts` The number of observations represented by each state. Defaults to 1 so that each state represents one observation Number in the interval [0, Inf], or a vector of these.

`dimensions` List of named lists or `go_parcats_dimension()` objects.

`domain` Named list or `go_parcats_domain()` object.

`hoverinfo` Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of `"count"`, `"probability"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`.

`hoveron` Sets the hover interaction mode for the parcats diagram. If 'category', hover interaction take place per category. If 'color', hover interactions take place per color per category. If 'dimension', hover interactions take place across all categories per dimension. One of `"category"`, `"color"`, `"dimension"`.

`hovertemplate` `go_parcats()`: Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'count', 'probability', 'category', 'categorycount', 'colorcount' and 'bandcolorcount'. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String.

`go_parcats_line()`: Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will

be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'count' and 'probability'. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`labelfont`
: Sets the font for the 'dimension' labels. Named list or `go_parcats_labelfont()` object.

`legendgrouptitle`
: Named list or `go_parcats_legendgrouptitle()` object.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line`
: Named list or `go_parcats_line()` object.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

`name`
: `go_parcats()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_parcats_line_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

sortpaths — Sets the path sorting algorithm. If 'forward', sort paths based on dimension categories from left to right. If 'backward', sort paths based on dimensions categories from right to left. One of "forward", "backward".

tickfont — go_parcats(): Sets the font for the 'category' labels. Named list or go_parcats_tickfont() object.

go_parcats_line_colorbar(): Sets the color bar's tick label font Named list or go_parcats_line_colorbar_tickfont() object.

uid — Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision — Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

visible — go_parcats(): Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.

go_parcats_dimension(): Shows the dimension when set to 'true' (the default). Hides the dimension for 'false'. Logical.

row — add_parcats(): Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.

go_parcats_domain(): If there is a layout grid, use the domain for this row in the grid for this parcats trace. Integer in the interval [0, Inf].

col — Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

skip_invalid — If TRUE, silently drop invalid properties instead of raising an error.

arg — Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

categoryarray — Sets the order in which categories in this dimension appear. Only has an effect if 'categoryorder' is set to "array". Used with 'categoryorder'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

categoryorder — Specifies the ordering logic for the categories in the dimension. By default, plotly uses "trace", which specifies the order that is present in the data supplied. Set 'categoryorder' to "category ascending" or "category descending" if order should be determined by the alphanumerical order of the category names. Set 'categoryorder' to "array" to derive the ordering from the attribute 'categoryarray'. If a category is not found in the 'categoryarray' array, the sorting behavior for that attribute will be identical to the "trace" mode. The unspecified categories will follow the categories in 'categoryarray'. One of "trace", "category ascending", "category descending", "array".

displayindex — The display index of dimension, from left to right, zero indexed, defaults to dimension index. Integer.

label — The shown name of the dimension. String.

ticktext — go_parcats_dimension(): Sets alternative tick labels for the categories in this dimension. Only has an effect if ‘categoryorder‘ is set to "array". Should be an array the same length as ‘categoryarray‘ Used with ‘categoryorder‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_parcats_line_colorbar(): Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

values — Dimension values. ‘values[n]‘ represents the category value of the ‘n‘th point in the dataset, therefore the ‘values‘ vector for all dimensions must be the same (longer vectors will be truncated). Data array: a vector, factor, Date/POSIXct vector, matrix or list.

column — If there is a layout grid, use the domain for this column in the grid for this parcats trace. Integer in the interval [0, Inf].

x — go_parcats_domain(): Sets the horizontal domain of this parcats trace (in plot fraction). Vector or list of 2 element(s).

go_parcats_line_colorbar(): Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

y — go_parcats_domain(): Sets the vertical domain of this parcats trace (in plot fraction). Vector or list of 2 element(s).

go_parcats_line_colorbar(): Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

color — go_parcats_labelfont(), go_parcats_legendgrouptitle_font(), go_parcats_line_colorbar_t
go_parcats_line_colorbar_title_font(), go_parcats_tickfont(): Color: a CSS color string or R color name.

go_parcats_line(): Sets the line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘line.cmin‘ and ‘line.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by parcats.line.colorscale, or a vector of these.

family — HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String.

lineposition	Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".
shadow	Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.
size	Number in the interval [1, Inf].
style	Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".
textcase	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".
variant	Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
weight	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
font	go_parcats_legendgrouptitle(): Sets this legend group's title font. Named list or go_parcats_legendgrouptitle_font() object. go_parcats_line_colorbar_title(): Sets this color bar's title font. Named list or go_parcats_line_colorbar_title_font() object.
text	go_parcats_legendgrouptitle(): Sets the title of the legend group. String. go_parcats_line_colorbar_title(): Sets the title of the color bar. String.
autocolorscale	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘line.colorscale‘. Has an effect only if in ‘line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
cauto	Determines whether or not the color domain is computed with respect to the input data (here in ‘line.color‘) or the bounds set in ‘line.cmin‘ and ‘line.cmax‘ Has an effect only if in ‘line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘line.cmin‘ and ‘line.cmax‘ are set by the user. Logical.
cmax	Sets the upper bound of the color domain. Has an effect only if in ‘line.color‘ is set to a numerical array. Value should have the same units as in ‘line.color‘ and if set, ‘line.cmin‘ must be set as well. Number.
cmid	Sets the mid-point of the color domain by scaling ‘line.cmin‘ and/or ‘line.cmax‘ to be equidistant to this point. Has an effect only if in ‘line.color‘ is set to a numerical array. Value should have the same units as in ‘line.color‘. Has no effect when ‘line.cauto‘ is ‘false‘. Number.
cmin	Sets the lower bound of the color domain. Has an effect only if in ‘line.color‘ is set to a numerical array. Value should have the same units as in ‘line.color‘ and if set, ‘line.cmax‘ must be set as well. Number.
coloraxis	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: "coloraxis", "coloraxis2", "coloraxis3", ...

colorbar	Named list or go_parcats_line_colorbar() object.
colorscale	Sets the colorscale. Has an effect only if in 'line.color' is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the bounds of the colorscale in color space, use 'line.cmin' and 'line.cmax'. Alternatively, 'colorscale' may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,I Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).
reversescale	Reverses the color mapping if true. Has an effect only if in 'line.color' is set to a numerical array. If true, 'line.cmin' will correspond to the last color in the array and 'line.cmax' will correspond to the first color. Logical.
shape	Sets the shape of the paths. If 'linear', paths are composed of straight lines. If 'hspline', paths are composed of horizontal curved splines One of "linear", "hspline".
showscale	Determines whether or not a colorbar is displayed for this trace. Has an effect only if in 'line.color' is set to a numerical array. Logical.
bgcolor	Sets the color of padded area. Color: a CSS color string or R color name.
bordercolor	Sets the axis line color. Color: a CSS color string or R color name.
borderwidth	Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].
dtick	Sets the step in-between ticks on this color bar. Use with 'tick0'. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.
exponentformat	Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".
labelalias	Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.
len	Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops
: List of named lists or go_parcats_line_colorbar_tickformatstop() objects.

ticklabeloverflow
: Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelstep
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

ticklen
: Sets the tick length (in px). Number in the interval [0, Inf].

tickmode
: Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of "auto", "linear", "array".

tickprefix
: Sets a tick label prefix. String.

ticks
: Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of "outside", "inside", "".

ticksuffix
: Sets a tick label suffix. String.

tickvals
: Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth
: Sets the tick width (in px). Number in the interval [0, Inf].

title
: String (the title text), named list or go_parcats_line_colorbar_title() object.

xanchor
: Sets this color bar's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the color bar. Defaults to "left" when 'orientation' is "v" and "center" when 'orientation' is "h". One of "left", "center", "right".

xpad
: Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".

yanchor — Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

side — Determines the location of color bar’s title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of "right", "top", "bottom".

Details

go_parcats() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_parcats() creates an object and adds it to a figure.

Nested objects documented here:

go_parcats_dimension() parcats.dimension: The dimensions (variables) of the parallel categories diagram.

go_parcats_domain() parcats.domain

go_parcats_labelfont() parcats.labelfont: Sets the font for the ‘dimension‘ labels.

go_parcats_legendgrouptitle() parcats.legendgrouptitle

go_parcats_legendgrouptitle_font() parcats.legendgrouptitle.font: Sets this legend group’s title font.

go_parcats_line() parcats.line

go_parcats_line_colorbar() parcats.line.colorbar

`go_parcats_line_colorbar_tickfont()` `parcats.line.colorbar.tickfont`: Sets the color bar's tick label font

`go_parcats_line_colorbar_tickformatstop()` `parcats.line.colorbar.tickformatstop`

`go_parcats_line_colorbar_title()` `parcats.line.colorbar.title`

`go_parcats_line_colorbar_title_font()` `parcats.line.colorbar.title.font`: Sets this color bar's title font.

`go_parcats_tickfont()` `parcats.tickfont`: Sets the font for the 'category' labels.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_parcats()
```

`add_parcoords` *Parcoords traces*

Description

Parallel coordinates for multidimensional exploratory data analysis. The samples are specified in 'dimensions'. The colors are set in 'line.color'.

Usage

```
add_parcoords(fig, ...)

## Default S3 method:
add_parcoords(
  fig,
  ...,
  data = NULL,
  customdata = NULL,
  dimensions = NULL,
  domain = NULL,
  ids = NULL,
  labelangle = NULL,
  labelfont = NULL,
  labelside = NULL,
  legend = NULL,
```

```
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  meta = NULL,
  name = NULL,
  rangefont = NULL,
  tickfont = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_parcoords(
  arg = NULL,
  ...,
  customdata = NULL,
  dimensions = NULL,
  domain = NULL,
  ids = NULL,
  labelangle = NULL,
  labelfont = NULL,
  labelside = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  meta = NULL,
  name = NULL,
  rangefont = NULL,
  tickfont = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_parcoords_dimension(
  arg = NULL,
  ...,
  constraintrange = NULL,
  label = NULL,
```

```
  multiselect = NULL,
  name = NULL,
  range = NULL,
  templateitemname = NULL,
  tickformat = NULL,
  ticktext = NULL,
  tickvals = NULL,
  values = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_parcoords_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_parcoords_labelfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_parcoords_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_parcoords_legendgrouptitle_font(
  arg = NULL,
  ...,
```

```
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_parcoords_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  reversescale = NULL,
  showscale = NULL,
  skip_invalid = FALSE
)

go_parcoords_line_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
```

```
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_parcoords_line_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

```
  go_parcoords_line_colorbar_tickformatstop(
    arg = NULL,
    ...,
    dtickrange = NULL,
    enabled = NULL,
    name = NULL,
    templateitemname = NULL,
    value = NULL,
    skip_invalid = FALSE
  )

  go_parcoords_line_colorbar_title(
    arg = NULL,
    ...,
    font = NULL,
    side = NULL,
    text = NULL,
    skip_invalid = FALSE
  )

  go_parcoords_line_colorbar_title_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_parcoords_rangefont(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
```

```
)

go_parcoords_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_parcoords_unselected(arg = NULL, ..., line = NULL, skip_invalid = FALSE)

go_parcoords_unselected_line(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
dimensions	List of named lists or go_parcoords_dimension() objects.
domain	Named list or go_parcoords_domain() object.
ids	Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`labelangle` Sets the angle of the labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the labels vertically. Tilted labels with "labelangle" may be positioned better inside margins when ‘labelposition‘ is set to "bottom". Angle in degrees.

`labelfont` Sets the font for the ‘dimension‘ labels. Named list or `go_parcoords_labelfont()` object.

`labelside` Specifies the location of the ‘label‘. "top" positions labels above, next to the title "bottom" positions labels below the graph Tilted labels with "labelangle" may be positioned better inside margins when ‘labelposition‘ is set to "bottom". One of `"top"`, `"bottom"`.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgrouptitle`
Named list or `go_parcoords_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line` `go_parcoords()`: Named list or `go_parcoords_line()` object.
`go_parcoords_unselected()`: Named list or `go_parcoords_unselected_line()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_parcoords()`: Sets the trace name. The trace name appears as the legend item and on hover. String.
`go_parcoords_dimension()`,`go_parcoords_line_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`rangefont` Sets the font for the ‘dimension‘ range values. Named list or `go_parcoords_rangefont()` object.

tickfont
: go_parcoords(): Sets the font for the ‘dimension‘ tick values. Named list or go_parcoords_tickfont() object.
go_parcoords_line_colorbar(): Sets the color bar's tick label font Named list or go_parcoords_line_colorbar_tickfont() object.

uid
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision
: Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

unselected
: Named list or go_parcoords_unselected() object.

visible
: go_parcoords(): Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.
go_parcoords_dimension(): Shows the dimension when set to ‘true‘ (the default). Hides the dimension for ‘false‘. Logical.

row
: add_parcoords(): Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
go_parcoords_domain(): If there is a layout grid, use the domain for this row in the grid for this parcoords trace. Integer in the interval [0, Inf].

col
: Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

skip_invalid
: If TRUE, silently drop invalid properties instead of raising an error.

arg
: Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

constraintrange
: The domain range to which the filter on the dimension is constrained. Must be an array of ‘[fromValue, toValue]‘ with ‘fromValue <= toValue‘, or if ‘multiselect‘ is not disabled, you may give an array of arrays, where each inner array is ‘[fromValue, toValue]‘. Vector or list of up to 2 element(s).

label
: The shown name of the dimension. String.

multiselect
: Do we allow multiple selection ranges or just a single range? Logical.

range
: The domain range that represents the full, shown axis extent. Defaults to the ‘values‘ extent. Must be an array of ‘[fromValue, toValue]‘ with finite numbers as elements. Vector or list of 2 element(s).

templateitemname
: Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure,

but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`ticktext` `go_parcoords_dimension()`: Sets the text displayed at the ticks position via ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_parcoords_line_colorbar()`: Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` `go_parcoords_dimension()`: Sets the values at which ticks on this axis appear. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_parcoords_line_colorbar()`: Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`values` Dimension values. ‘values[n]‘ represents the value of the ‘n‘th point in the dataset, therefore the ‘values‘ vector for all dimensions must be the same (longer vectors will be truncated). Each value must be a finite number. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`column` If there is a layout grid, use the domain for this column in the grid for this parcoords trace. Integer in the interval [0, Inf].

`x` `go_parcoords_domain()`: Sets the horizontal domain of this parcoords trace (in plot fraction). Vector or list of 2 element(s).

`go_parcoords_line_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`y` `go_parcoords_domain()`: Sets the vertical domain of this parcoords trace (in plot fraction). Vector or list of 2 element(s).

`go_parcoords_line_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

`color` `go_parcoords_labelfont()`, `go_parcoords_legendgrouptitle_font()`, `go_parcoords_line_colo`
`go_parcoords_line_colorbar_title_font()`, `go_parcoords_rangefont()`,
`go_parcoords_tickfont()`: Color: a CSS color string or R color name.

	`go_parcoords_line()`: Sets the line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'line.cmin' and 'line.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by `parcoords.line.colorscale`, or a vector of these. `go_parcoords_unselected_line()`: Sets the base color of unselected lines. in connection with 'unselected.line.opacity'. Color: a CSS color string or R color name.
`family`	HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.
`lineposition`	Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.
`shadow`	Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.
`size`	Number in the interval [1, Inf].
`style`	Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.
`textcase`	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.
`variant`	Sets the variant of the font. One of `"normal"`, `"small-caps"`, `"all-small-caps"`, `"all-petite-caps"`, `"petite-caps"`, `"unicase"`.
`weight`	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
`font`	`go_parcoords_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_parcoords_legendgrouptitle_font()` object. `go_parcoords_line_colorbar_title()`: Sets this color bar's title font. Named list or `go_parcoords_line_colorbar_title_font()` object.
`text`	`go_parcoords_legendgrouptitle()`: Sets the title of the legend group. String. `go_parcoords_line_colorbar_title()`: Sets the title of the color bar. String.
`autocolorscale`	Determines whether the colorscale is a default palette ('autocolorscale: true') or the palette determined by 'line.colorscale'. Has an effect only if in 'line.color' is set to a numerical array. In case 'colorscale' is unspecified or 'autocolorscale' is true, the default palette will be chosen according to whether numbers in the 'color' array are all positive, all negative or mixed. Logical.
`cauto`	Determines whether or not the color domain is computed with respect to the input data (here in 'line.color') or the bounds set in 'line.cmin' and 'line.cmax' Has an effect only if in 'line.color' is set to a numerical array. Defaults to 'false' when 'line.cmin' and 'line.cmax' are set by the user. Logical.
`cmax`	Sets the upper bound of the color domain. Has an effect only if in 'line.color' is set to a numerical array. Value should have the same units as in 'line.color' and if set, 'line.cmin' must be set as well. Number.

`cmid` Sets the mid-point of the color domain by scaling ‘line.cmin‘ and/or ‘line.cmax‘ to be equidistant to this point. Has an effect only if in ‘line.color‘ is set to a numerical array. Value should have the same units as in ‘line.color‘. Has no effect when ‘line.cauto‘ is ‘false‘. Number.

`cmin` Sets the lower bound of the color domain. Has an effect only if in ‘line.color‘ is set to a numerical array. Value should have the same units as in ‘line.color‘ and if set, ‘line.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_parcoords_line_colorbar()` object.

`colorscale` Sets the colorscale. Has an effect only if in ‘line.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘line.cmin‘ and ‘line.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,I Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`reversescale` Reverses the color mapping if true. Has an effect only if in ‘line.color‘ is set to a numerical array. If true, ‘line.cmin‘ will correspond to the last color in the array and ‘line.cmax‘ will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘line.color‘ is set to a numerical array. Logical.

`bgcolor` Sets the color of padded area. Color: a CSS color string or R color name.

`bordercolor` Sets the axis line color. Color: a CSS color string or R color name.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickformatstops
List of named lists or go_parcoords_line_colorbar_tickformatstop() objects.

ticklabeloverflow
Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition
Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelstep Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

ticklen Sets the tick length (in px). Number in the interval [0, Inf].

tickmode Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of "auto", "linear", "array".

tickprefix Sets a tick label prefix. String.

ticks Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of "outside", "inside", "".

ticksuffix Sets a tick label suffix. String.

tickwidth Sets the tick width (in px). Number in the interval [0, Inf].

title String (the title text), named list or go_parcoords_line_colorbar_title() object.

xanchor Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of "left", "center", "right".

xpad Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref	Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".
yanchor	Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of "top", "middle", "bottom".
ypad	Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].
yref	Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".
dtickrange	range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).
enabled	Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.
value	string - dtickformat for described zoom level, the same as "tickformat" String.
side	Determines the location of color bar’s title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of "right", "top", "bottom".
opacity	Sets the opacity of unselected lines. The default "auto" decreases the opacity smoothly as the number of lines increases. Use 1 to achieve exact ‘unselected.line.color‘. Number in the interval [0, 1].

Details

`go_parcoords()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_parcoords()` creates an object and adds it to a figure.

Nested objects documented here:

`go_parcoords_dimension()` `parcoords.dimension`: The dimensions (variables) of the parallel coordinates chart. 2..60 dimensions are supported.

`go_parcoords_domain()` `parcoords.domain`

`go_parcoords_labelfont()` `parcoords.labelfont`: Sets the font for the ‘dimension‘ labels.

`go_parcoords_legendgrouptitle()` `parcoords.legendgrouptitle`

`go_parcoords_legendgrouptitle_font()` `parcoords.legendgrouptitle.font`: Sets this legend group’s title font.

`go_parcoords_line()` `parcoords.line`

`go_parcoords_line_colorbar()` `parcoords.line.colorbar`

`go_parcoords_line_colorbar_tickfont()` `parcoords.line.colorbar.tickfont`: Sets the color bar’s tick label font

`go_parcoords_line_colorbar_tickformatstop()` `parcoords.line.colorbar.tickformatstop`

`go_parcoords_line_colorbar_title()` `parcoords.line.colorbar.title`

`go_parcoords_line_colorbar_title_font()` `parcoords.line.colorbar.title.font`: Sets this color bar's title font.

`go_parcoords_rangefont()` `parcoords.rangefont`: Sets the font for the 'dimension' range values.

`go_parcoords_tickfont()` `parcoords.tickfont`: Sets the font for the 'dimension' tick values.

`go_parcoords_unselected()` `parcoords.unselected`

`go_parcoords_unselected_line()` `parcoords.unselected.line`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_parcoords()
```

---

add_pie *Pie traces*

Description

A data visualized by the sectors of the pie is set in 'values'. The sector labels are set in 'labels'. The sector colors are set in 'marker.colors'

Usage

```
add_pie(fig, ...)

## Default S3 method:
add_pie(
  fig,
  ...,
  data = NULL,
  automargin = NULL,
  customdata = NULL,
  direction = NULL,
  dlabel = NULL,
  domain = NULL,
  hole = NULL,
  hoverinfo = NULL,
```

```
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextfont = NULL,
  insidetextorientation = NULL,
  label0 = NULL,
  labels = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  outsidetextfont = NULL,
  pull = NULL,
  rotation = NULL,
  scalegroup = NULL,
  showlegend = NULL,
  sort = NULL,
  text = NULL,
  textfont = NULL,
  textinfo = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  title = NULL,
  uid = NULL,
  uirevision = NULL,
  values = NULL,
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_pie(
  arg = NULL,
  ...,
  automargin = NULL,
  customdata = NULL,
  direction = NULL,
  dlabel = NULL,
  domain = NULL,
```

```
  hole = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextfont = NULL,
  insidetextorientation = NULL,
  label0 = NULL,
  labels = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  outsidetextfont = NULL,
  pull = NULL,
  rotation = NULL,
  scalegroup = NULL,
  showlegend = NULL,
  sort = NULL,
  text = NULL,
  textfont = NULL,
  textinfo = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  title = NULL,
  uid = NULL,
  uirevision = NULL,
  values = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_pie_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
```

```
)

go_pie_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_pie_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_pie_insidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_pie_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
```

```
  text = NULL,
  skip_invalid = FALSE
)

go_pie_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_pie_marker(
  arg = NULL,
  ...,
  colors = NULL,
  line = NULL,
  pattern = NULL,
  skip_invalid = FALSE
)

go_pie_marker_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_pie_marker_pattern(
  arg = NULL,
  ...,
  bgcolor = NULL,
  fgcolor = NULL,
  fgopacity = NULL,
  fillmode = NULL,
  path = NULL,
  shape = NULL,
  size = NULL,
  solidity = NULL,
  skip_invalid = FALSE
```

```
)

go_pie_outsidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_pie_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_pie_title(
  arg = NULL,
  ...,
  font = NULL,
  position = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_pie_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
```

```
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
automargin	Determines whether outside text labels can push the margins. Logical.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
direction	Specifies the direction at which succeeding sectors follow one another. One of "clockwise", "counterclockwise".
dlabel	Sets the label step. See 'label0' for more info. Number.
domain	Named list or go_pie_domain() object.
hole	Sets the fraction of the radius to cut out of the pie. Use this to make a donut chart. Number in the interval [0, 1].
hoverinfo	Determines what trace information appears on hover. Flags are rendered in a fixed order; use 'hovertemplate' if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of "label", "text", "value", "percent", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_pie_hoverlabel() object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for

details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'label', 'color', 'value', 'percent' and 'text'. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each sector. If a single string, the same string appears for all data points. If an array of string, the items are mapped in order of this trace's sectors. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`insidetextfont`
: Sets the font used for 'textinfo' lying inside the sector. Named list or `go_pie_insidetextfont()` object.

`insidetextorientation`
: Controls the orientation of the text inside chart sectors. When set to "auto", text may be oriented in any direction in order to be as big as possible in the middle of a sector. The "horizontal" option orients text to be parallel with the bottom of the chart, and may make text smaller in order to achieve that goal. The "radial" option orients text along the radius of the sector. The "tangential" option orients text perpendicular to the radius of the sector. One of `"horizontal"`, `"radial"`, `"tangential"`, `"auto"`.

`label0`
: Alternate to 'labels'. Builds a numeric set of labels. Use with 'dlabel' where 'label0' is the starting label and 'dlabel' the step. Number.

`labels`
: Sets the sector labels. If 'labels' entries are duplicated, we sum associated 'values' or simply count occurrences if 'values' is not provided. For other array attributes (including color) we use the first non-empty entry among all occurrences of the label. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show the pie slices in. Can be an array if 'values' is set. In that case, each entry specifies the legend reference for one slice. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_pie_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this pie. If passed as an array, this will set the legend rank of the individual pie slices. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number, or a vector of these.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`marker`
: Named list or `go_pie_marker()` object.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name`
: Sets the trace name. The trace name appears as the legend item and on hover. String.

`opacity`
: Sets the opacity of the trace. Number in the interval [0, 1].

`outsidetextfont`
: Sets the font used for ‘textinfo‘ lying outside the sector. Named list or `go_pie_outsidetextfont()` object.

`pull`
: Sets the fraction of larger radius to pull the sectors out from the center. This can be a constant to pull all slices apart from each other equally or an array to highlight one or more slices. Number in the interval [0, 1], or a vector of these.

`rotation`
: Instead of the first slice starting at 12 o’clock, rotate to some other angle. Angle in degrees.

`scalegroup`
: If there are multiple pie charts that should be sized according to their totals, link them by providing a non-empty group id here shared by every trace in the same group. String.

`showlegend`
: Determines whether or not items corresponding to the pie slices are shown in the legend. Can be an array if ‘values‘ is set. In that case, each entry specifies appearance in the legend for one slice. Logical, or a vector of these.

`sort`
: Determines whether or not the sectors are reordered from largest to smallest. Logical.

`text`
: `go_pie()`: Sets text elements associated with each sector. If trace ‘textinfo‘ contains a "text" flag, these elements will be seen on the chart. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_pie_legendgrouptitle(): Sets the title of the legend group. String.
go_pie_title(): Sets the title of the chart. If it is empty, no title is displayed. String.

textfont — Sets the font used for 'textinfo'. Named list or go_pie_textfont() object.

textinfo — Determines what trace information appears on the graph. Flags are rendered in a fixed order; use 'texttemplate' if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of "label", "text", "value", "percent" joined with "+", or one of "none".

textposition — Specifies the location of the 'textinfo'. One of "inside", "outside", "auto", "none", or a vector of these.

texttemplate — Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'label', 'color', 'value', 'percent' and 'text'. String, or a vector of these.

texttemplatefallback — Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

title — String (the title text), named list or go_pie_title() object.

uid — Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision — Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

values — Sets the values of the sectors. If omitted, we count occurrences of each label. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

visible — Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.

row	add_pie(): Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells. go_pie_domain(): If there is a layout grid, use the domain for this row in the grid for this pie trace. Integer in the interval [0, Inf].
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
column	If there is a layout grid, use the domain for this column in the grid for this pie trace. Integer in the interval [0, Inf].
x	Sets the horizontal domain of this pie trace (in plot fraction). Vector or list of 2 element(s).
y	Sets the vertical domain of this pie trace (in plot fraction). Vector or list of 2 element(s).
align	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.
bgcolor	go_pie_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these. go_pie_marker_pattern(): When there is no colorscale sets the color of background pattern fill. Defaults to a 'marker.color' background when 'fillmode' is "overlay". Otherwise, defaults to a transparent background. Color: a CSS color string or R color name, or a vector of these.
bordercolor	Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.
font	go_pie_hoverlabel(): Sets the font used in hover labels. Named list or go_pie_hoverlabel_font() object. go_pie_legendgrouptitle(): Sets this legend group's title font. Named list or go_pie_legendgrouptitle_font() object. go_pie_title(): Sets the font used for 'title'. Named list or go_pie_title_font() object.
namelength	Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.
showarrow	Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.
color	go_pie_hoverlabel_font(), go_pie_insidetextfont(), go_pie_outsidetextfont(), go_pie_textfont(), go_pie_title_font(): Color: a CSS color string or R color name, or a vector of these.

go_pie_legendgrouptitle_font(): Color: a CSS color string or R color name.

go_pie_marker_line(): Sets the color of the line enclosing each sector. Color: a CSS color string or R color name, or a vector of these.

family — go_pie_hoverlabel_font(), go_pie_insidetextfont(), go_pie_outsidetextfont(), go_pie_textfont(), go_pie_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

go_pie_legendgrouptitle_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition — go_pie_hoverlabel_font(), go_pie_insidetextfont(), go_pie_outsidetextfont(), go_pie_textfont(), go_pie_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_pie_legendgrouptitle_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow — go_pie_hoverlabel_font(), go_pie_insidetextfont(), go_pie_outsidetextfont(), go_pie_textfont(), go_pie_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_pie_legendgrouptitle_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size — go_pie_hoverlabel_font(), go_pie_insidetextfont(), go_pie_outsidetextfont(), go_pie_textfont(), go_pie_title_font(): Number in the interval [1, Inf], or a vector of these.

go_pie_legendgrouptitle_font(): Number in the interval [1, Inf].

go_pie_marker_pattern(): Sets the size of unit squares of the pattern fill in pixels, which corresponds to the interval of repetition of the pattern. Number in the interval [0, Inf], or a vector of these.

style — go_pie_hoverlabel_font(), go_pie_insidetextfont(), go_pie_outsidetextfont(), go_pie_textfont(), go_pie_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_pie_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_pie_hoverlabel_font(), go_pie_insidetextfont(), go_pie_outsidetextfont(), go_pie_textfont(), go_pie_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.
go_pie_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_pie_hoverlabel_font(), go_pie_insidetextfont(), go_pie_outsidetextfont(), go_pie_textfont(), go_pie_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.
go_pie_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_pie_hoverlabel_font(), go_pie_insidetextfont(), go_pie_outsidetextfont(), go_pie_textfont(), go_pie_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.
go_pie_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

colors
: Sets the color of each sector. If not specified, the default trace color set is used to pick the sector colors. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

line
: Named list or go_pie_marker_line() object.

pattern
: Sets the pattern within the marker. Named list or go_pie_marker_pattern() object.

width
: Sets the width (in px) of the line enclosing each sector. Number in the interval [0, Inf], or a vector of these.

fgcolor
: When there is no colorscale sets the color of foreground pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "replace". Otherwise, defaults to dark grey or white to increase contrast with the ‘bgcolor‘. Color: a CSS color string or R color name, or a vector of these.

fgopacity
: Sets the opacity of the foreground pattern fill. Defaults to a 0.5 when ‘fillmode‘ is "overlay". Otherwise, defaults to 1. Number in the interval [0, 1].

fillmode
: Determines whether ‘marker.color‘ should be used as a default to ‘bgcolor‘ or a ‘fgcolor‘. One of "replace", "overlay".

path
: Sets a custom path for pattern fill. Use with no ‘shape‘ or ‘solidity‘, provide an SVG path string for the regions of the square from (0,0) to (‘size‘,‘size‘) to color. String, or a vector of these.

shape
: Sets the shape of the pattern fill. By default, no pattern is used for filling the area. One of "", "/", "\\", "x", "-", "|", "+", ".", or a vector of these.

solidity
: Sets the solidity of the pattern fill. Solidity is roughly the fraction of the area filled by the pattern. Solidity of 0 shows only the background color without pattern and solidty of 1 shows only the foreground color without pattern. Number in the interval [0, 1], or a vector of these.

position
: Specifies the location of the ‘title‘. One of "top left", "top center", "top right", "middle center", "bottom left", "bottom center", "bottom right".

Details

`go_pie()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_pie()` creates an object and adds it to a figure.

Nested objects documented here:

`go_pie_domain()` `pie.domain`

`go_pie_hoverlabel()` `pie.hoverlabel`

`go_pie_hoverlabel_font()` `pie.hoverlabel.font`: Sets the font used in hover labels.

`go_pie_insidetextfont()` `pie.insidetextfont`: Sets the font used for 'textinfo' lying inside the sector.

`go_pie_legendgrouptitle()` `pie.legendgrouptitle`

`go_pie_legendgrouptitle_font()` `pie.legendgrouptitle.font`: Sets this legend group's title font.

`go_pie_marker()` `pie.marker`

`go_pie_marker_line()` `pie.marker.line`

`go_pie_marker_pattern()` `pie.marker.pattern`: Sets the pattern within the marker.

`go_pie_outsidetextfont()` `pie.outsidetextfont`: Sets the font used for 'textinfo' lying outside the sector.

`go_pie_textfont()` `pie.textfont`: Sets the font used for 'textinfo'.

`go_pie_title()` `pie.title`

`go_pie_title_font()` `pie.title.font`: Sets the font used for 'title'.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_pie()
```

add_quiver *Quiver traces*

Description

The quiver trace type visualizes vector fields using arrows. Specify a vector field using 4 1D arrays: 2 position arrays ‘x‘, ‘y‘ and 2 vector component arrays ‘u‘, ‘v‘. The arrows are drawn exactly at the positions given by ‘x‘ and ‘y‘. Arrow length and direction are determined by ‘u‘ and ‘v‘ components.

Usage

```
add_quiver(fig, ...)

## Default S3 method:
add_quiver(
  fig,
  ...,
  data = NULL,
  anchor = NULL,
  arrowref = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lengthfactor = NULL,
  lengthmode = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  u = NULL,
  uhoverformat = NULL,
```

```
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  v = NULL,
  vhoverformat = NULL,
  visible = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xhoverformat = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_quiver(
  arg = NULL,
  ...,
  anchor = NULL,
  arrowref = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lengthfactor = NULL,
  lengthmode = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textfont = NULL,
```

```
  textposition = NULL,
  u = NULL,
  uhoverformat = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  v = NULL,
  vhoverformat = NULL,
  visible = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xhoverformat = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  skip_invalid = FALSE
)

go_quiver_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_quiver_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_quiver_legendgrouptitle(
```

```
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_quiver_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_quiver_marker(
  arg = NULL,
  ...,
  arrowsize = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  line = NULL,
  reversescale = NULL,
  showscale = NULL,
  skip_invalid = FALSE
)

go_quiver_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
```

```
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_quiver_marker_colorbar_tickfont(
  arg = NULL,
  ...,
```

```
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_quiver_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_quiver_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_quiver_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_quiver_marker_line(
```

```
  arg = NULL,
  ...,
  dash = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_quiver_selected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_quiver_selected_marker(
  arg = NULL,
  ...,
  color = NULL,
  line = NULL,
  skip_invalid = FALSE
)

go_quiver_selected_marker_line(
  arg = NULL,
  ...,
  width = NULL,
  skip_invalid = FALSE
)

go_quiver_selected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

go_quiver_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
```

```
  weight = NULL,
  skip_invalid = FALSE
)

go_quiver_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_quiver_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  line = NULL,
  skip_invalid = FALSE
)

go_quiver_unselected_marker_line(
  arg = NULL,
  ...,
  width = NULL,
  skip_invalid = FALSE
)

go_quiver_unselected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
anchor	Sets the vector arrows' anchor with respect to their (x,y) positions. Use "tail" to place (x,y) at the base, "tip" to place (x,y) at the head, or "center" to center the vector arrow on (x,y). One of "tip", "tail", "center".

`arrowref` Determines how the u/v vector components are interpreted. If "paper", u/v are interpreted in pixel coordinates and the rendered vector angle does not change regardless of the axis scales. If "data", u/v are interpreted in data coordinates and the rendered vector angle may change, e.g. if zooming in along a single axis One of `"paper"`, `"data"`.

`customdata` Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`dx` Sets the x coordinate step. See ‘x0‘ for more info. Number.

`dy` Sets the y coordinate step. See ‘y0‘ for more info. Number.

`hoverinfo` Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"u"`, `"v"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_quiver_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variables ‘x‘, ‘y‘, ‘u‘, ‘v‘, ‘text‘ and ‘name‘. Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example ‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
Named list or `go_quiver_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`lengthfactor` Adjusts the drawn length of the vector arrows. The arrow length is determined by the values of u and v, then optionally rescaled when ‘lengthmode‘ is "scaled", then multiplied by ‘lengthfactor‘. Number in the interval [0, Inf].

`lengthmode` Determines whether vector arrows are drawn according to their raw lengths, or scaled based on the maximum vector length and point density. Note: When ‘arrowref‘ is "paper" vectors are always scaled and ‘lengthmode‘ "raw" is ignored. One of `"scaled"`, `"raw"`.

`marker` `go_quiver()`: Named list or `go_quiver_marker()` object.

`go_quiver_selected()`: Named list or `go_quiver_selected_marker()` object.

`go_quiver_unselected()`: Named list or `go_quiver_unselected_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_quiver()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_quiver_marker_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` Sets the opacity of the trace. Number in the interval [0, 1].

`selected` Named list or `go_quiver_selected()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection

where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`text` `go_quiver()`: Sets text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_quiver_legendgrouptitle()`: Sets the title of the legend group. String.

`go_quiver_marker_colorbar_title()`: Sets the title of the color bar. String.

`textfont` `go_quiver()`: Sets the text font. Named list or `go_quiver_textfont()` object.

`go_quiver_selected()`: Named list or `go_quiver_selected_textfont()` object.

`go_quiver_unselected()`: Named list or `go_quiver_unselected_textfont()` object.

`textposition` Sets the positions of the ‘text‘ elements with respects to the (x,y) coordinates. One of `"top left"`, `"top center"`, `"top right"`, `"middle left"`, `"middle center"`, `"middle right"`, `"bottom left"`, `"bottom center"`, `"bottom right"`, or a vector of these.

`u` Sets the x components of the vector arrows. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`uhoverformat` Sets the hover text formatting rule for ‘u‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`unselected` Named list or `go_quiver_unselected()` object.

`v` Sets the y components of the vector arrows. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

vhoverformat Sets the hover text formatting rule for ‘v‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

visible Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

x `go_quiver()`: Sets the x coordinates of the vector arrow locations. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_quiver_marker_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

x0 Alternate to ‘x‘. Builds a linear space of x coordinates. Use with ‘dx‘ where ‘x0‘ is the starting coordinate and ‘dx‘ the step. Any value.

xaxis Sets a reference between this trace’s x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

xhoverformat Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

y `go_quiver()`: Sets the y coordinates of the vector arrow locations. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_quiver_marker_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

y0 Alternate to ‘y‘. Builds a linear space of y coordinates. Use with ‘dy‘ where ‘y0‘ is the starting coordinate and ‘dy‘ the step. Any value.

yaxis Sets a reference between this trace’s y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...

yhoverformat Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional

	seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.
`row`	Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.
`col`	Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.
`secondary_y`	If `TRUE`, use the secondary y axis of the subplot.
`skip_invalid`	If `TRUE`, silently drop invalid properties instead of raising an error.
`arg`	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
`align`	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.
`bgcolor`	`go_quiver_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these. `go_quiver_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.
`bordercolor`	`go_quiver_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these. `go_quiver_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.
`font`	`go_quiver_hoverlabel()`: Sets the font used in hover labels. Named list or `go_quiver_hoverlabel_font()` object. `go_quiver_legendgrouptitle()`: Sets this legend group’s title font. Named list or `go_quiver_legendgrouptitle_font()` object. `go_quiver_marker_colorbar_title()`: Sets this color bar’s title font. Named list or `go_quiver_marker_colorbar_title_font()` object.
`namelength`	Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to ‘namelength - 3‘ characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.
`showarrow`	Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.
`color`	`go_quiver_hoverlabel_font()`, `go_quiver_textfont()`: Color: a CSS color string or R color name, or a vector of these. `go_quiver_legendgrouptitle_font()`, `go_quiver_marker_colorbar_tickfont()`, `go_quiver_marker_colorbar_title_font()`: Color: a CSS color string or R color name. `go_quiver_marker()`: Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.cmin‘ and ‘marker.cmax‘ if set.

Color: a CSS color string or R color name, or a number mapped to a color by quiver.marker.colorscale, or a vector of these.

go_quiver_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name.

go_quiver_selected_textfont(): Sets the text font color of selected points, applied only when a selection exists. Color: a CSS color string or R color name.

go_quiver_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

go_quiver_unselected_textfont(): Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

family

go_quiver_hoverlabel_font(), go_quiver_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

go_quiver_legendgrouptitle_font(), go_quiver_marker_colorbar_tickfont(), go_quiver_marker_colorbar_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition

go_quiver_hoverlabel_font(), go_quiver_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_quiver_legendgrouptitle_font(), go_quiver_marker_colorbar_tickfont(), go_quiver_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow

go_quiver_hoverlabel_font(), go_quiver_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_quiver_legendgrouptitle_font(), go_quiver_marker_colorbar_tickfont(), go_quiver_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size

go_quiver_hoverlabel_font(), go_quiver_textfont(): Number in the interval [1, Inf], or a vector of these.

go_quiver_legendgrouptitle_font(), go_quiver_marker_colorbar_tickfont(), go_quiver_marker_colorbar_title_font(): Number in the interval [1, Inf].

style

go_quiver_hoverlabel_font(), go_quiver_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

	go_quiver_legendgrouptitle_font(), go_quiver_marker_colorbar_tickfont(), go_quiver_marker_colorbar_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".
textcase	go_quiver_hoverlabel_font(), go_quiver_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.
	go_quiver_legendgrouptitle_font(), go_quiver_marker_colorbar_tickfont(), go_quiver_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".
variant	go_quiver_hoverlabel_font(), go_quiver_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.
	go_quiver_legendgrouptitle_font(), go_quiver_marker_colorbar_tickfont(), go_quiver_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
weight	go_quiver_hoverlabel_font(), go_quiver_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.
	go_quiver_legendgrouptitle_font(), go_quiver_marker_colorbar_tickfont(), go_quiver_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
arrowsize	Sets the size of the vector arrowhead relative to 'marker.line.width'. A value of 1 (default) gives a head about 3x as wide as the line. Number in the interval [0.3, Inf].
autocolorscale	Determines whether the colorscale is a default palette ('autocolorscale: true') or the palette determined by 'marker.colorscale'. Has an effect only if in 'marker.color' is set to a numerical array. In case 'colorscale' is unspecified or 'autocolorscale' is true, the default palette will be chosen according to whether numbers in the 'color' array are all positive, all negative or mixed. Logical.
cauto	Determines whether or not the color domain is computed with respect to the input data (here in 'marker.color') or the bounds set in 'marker.cmin' and 'marker.cmax' Has an effect only if in 'marker.color' is set to a numerical array. Defaults to 'false' when 'marker.cmin' and 'marker.cmax' are set by the user. Logical.
cmax	Sets the upper bound of the color domain. Has an effect only if in 'marker.color' is set to a numerical array. Value should have the same units as in 'marker.color' and if set, 'marker.cmin' must be set as well. Number.
cmid	Sets the mid-point of the color domain by scaling 'marker.cmin' and/or 'marker.cmax' to be equidistant to this point. Has an effect only if in 'marker.color' is set to a numerical array. Value should have the same units as in 'marker.color'. Has no effect when 'marker.cauto' is 'false'. Number.
cmin	Sets the lower bound of the color domain. Has an effect only if in 'marker.color' is set to a numerical array. Value should have the same units as in 'marker.color' and if set, 'marker.cmax' must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_quiver_marker_colorbar()` object.

`colorscale` Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`line` `go_quiver_marker()`: Named list or `go_quiver_marker_line()` object.

`go_quiver_selected_marker()`: Named list or `go_quiver_selected_marker_line()` object.

`go_quiver_unselected_marker()`: Named list or `go_quiver_unselected_marker_line()` object.

`reversescale` Reverses the color mapping if true. Has an effect only if in ‘marker.color‘ is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘marker.color‘ is set to a numerical array. Logical.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1.

labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_quiver_marker_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops` List of named lists or `go_quiver_marker_colorbar_tickformatstop()` objects.

`ticklabeloverflow` Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition` Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals — Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth — Sets the tick width (in px). Number in the interval [0, Inf].

title — String (the title text), named list or go_quiver_marker_colorbar_title() object.

xanchor — Sets this color bar's horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of "left", "center", "right".

xpad — Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".

yanchor — Sets this color bar's vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

side — Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of "right", "top", "bottom".

dash — Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$".

`width` `go_quiver_marker_line()`: Sets the width (in px) of the vector arrow lines. Number in the interval [0, Inf].

`go_quiver_selected_marker_line()`: Sets the line width of selected points. Number in the interval [0, Inf].

`go_quiver_unselected_marker_line()`: Sets the line width of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

Details

`go_quiver()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_quiver()` creates an object and adds it to a figure.

Nested objects documented here:

`go_quiver_hoverlabel()` `quiver.hoverlabel`

`go_quiver_hoverlabel_font()` `quiver.hoverlabel.font`: Sets the font used in hover labels.

`go_quiver_legendgrouptitle()` `quiver.legendgrouptitle`

`go_quiver_legendgrouptitle_font()` `quiver.legendgrouptitle.font`: Sets this legend group's title font.

`go_quiver_marker()` `quiver.marker`

`go_quiver_marker_colorbar()` `quiver.marker.colorbar`

`go_quiver_marker_colorbar_tickfont()` `quiver.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_quiver_marker_colorbar_tickformatstop()` `quiver.marker.colorbar.tickformatstop`

`go_quiver_marker_colorbar_title()` `quiver.marker.colorbar.title`

`go_quiver_marker_colorbar_title_font()` `quiver.marker.colorbar.title.font`: Sets this color bar's title font.

`go_quiver_marker_line()` `quiver.marker.line`

`go_quiver_selected()` `quiver.selected`

`go_quiver_selected_marker()` `quiver.selected.marker`

`go_quiver_selected_marker_line()` `quiver.selected.marker.line`

`go_quiver_selected_textfont()` `quiver.selected.textfont`

`go_quiver_textfont()` `quiver.textfont`: Sets the text font.

`go_quiver_unselected()` `quiver.unselected`

`go_quiver_unselected_marker()` `quiver.unselected.marker`

`go_quiver_unselected_marker_line()` `quiver.unselected.marker.line`

`go_quiver_unselected_textfont()` `quiver.unselected.textfont`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_quiver(x = 1:3, y = c(2, 1, 3), marker_color = "red")
```

---

add_sankey *Sankey traces*

Description

Sankey plots for network flow data analysis. The nodes are specified in 'nodes' and the links between sources and targets in 'links'. The colors are set in 'nodes[i].color' and 'links[i].color', otherwise defaults are used.

Usage

```
add_sankey(fig, ...)

## Default S3 method:
add_sankey(
  fig,
  ...,
  data = NULL,
  arrangement = NULL,
  customdata = NULL,
  direction = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  ids = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  link = NULL,
  meta = NULL,
  name = NULL,
  node = NULL,
  orientation = NULL,
  selectedpoints = NULL,
  textfont = NULL,
  uid = NULL,
  uirevision = NULL,
  valueformat = NULL,
  valuesuffix = NULL,
```

```
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_sankey(
  arg = NULL,
  ...,
  arrangement = NULL,
  customdata = NULL,
  direction = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  ids = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  link = NULL,
  meta = NULL,
  name = NULL,
  node = NULL,
  orientation = NULL,
  selectedpoints = NULL,
  textfont = NULL,
  uid = NULL,
  uirevision = NULL,
  valueformat = NULL,
  valuesuffix = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_sankey_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_sankey_hoverlabel(
  arg = NULL,
  ...,
```

```
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_sankey_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_sankey_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_sankey_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_sankey_link(
```

```
  arg = NULL,
  ...,
  arrowlen = NULL,
  color = NULL,
  colorscales = NULL,
  customdata = NULL,
  hovercolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  label = NULL,
  line = NULL,
  sort = NULL,
  source = NULL,
  target = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_sankey_link_concentrationscales(
  arg = NULL,
  ...,
  cmax = NULL,
  cmin = NULL,
  colorscale = NULL,
  label = NULL,
  name = NULL,
  templateitemname = NULL,
  skip_invalid = FALSE
)

go_sankey_link_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_sankey_link_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
```

```
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_sankey_link_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_sankey_node(
  arg = NULL,
  ...,
  align = NULL,
  color = NULL,
  customdata = NULL,
  groups = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  label = NULL,
  line = NULL,
  pad = NULL,
  sort = NULL,
  thickness = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_sankey_node_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
```

```
  showarrow = NULL,
  skip_invalid = FALSE
)

go_sankey_node_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_sankey_node_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_sankey_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

`fig`	A `go_figure` object.
`...`	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.

data
: A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).

arrangement
: If value is 'snap' (the default), the node arrangement is assisted by automatic snapping of elements to preserve space between nodes specified via 'nodepad'. If value is 'perpendicular', the nodes can only move along a line perpendicular to the flow. If value is 'freeform', the nodes can freely move on the plane. If value is 'fixed', the nodes are stationary. One of "snap", "perpendicular", "freeform", "fixed".

customdata
: go_sankey(): Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  go_sankey_link(): Assigns extra data to each link. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  go_sankey_node(): Assigns extra data to each node. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

direction
: Sets the direction of the flow along the 'orientation' axis. With 'forward' (the default), sources are on the left (horizontal) or top (vertical). With 'reversed', sources are on the right (horizontal) or bottom (vertical). One of "forward", "reversed".

domain
: Named list or go_sankey_domain() object.

hoverinfo
: go_sankey(): Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Note that this attribute is superseded by 'node.hoverinfo' and 'node.hoverinfo' for nodes and links respectively. Any combination of "" joined with "+", or one of "all", "none", "skip".

  go_sankey_link(): Determines what trace information appears when hovering links. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. One of "all", "none", "skip".

  go_sankey_node(): Determines what trace information appears when hovering nodes. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. One of "all", "none", "skip".

hoverlabel
: go_sankey(): Named list or go_sankey_hoverlabel() object.

  go_sankey_link(): Named list or go_sankey_link_hoverlabel() object.

  go_sankey_node(): Named list or go_sankey_node_hoverlabel() object.

ids
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

legend
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: "legend", "legend2", "legend3", ...

`legendgrouptitle`
: Named list or `go_sankey_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`link`
: The links of the Sankey plot. Named list or `go_sankey_link()` object.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name`
: `go_sankey()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

  `go_sankey_link_concentrationscales()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`node`
: The nodes of the Sankey plot. Named list or `go_sankey_node()` object.

`orientation`
: Sets the orientation of the Sankey diagram. With ‘h‘ (the default), the flow runs horizontally. With ‘v‘, the flow runs vertically. Use ‘direction‘ to control which side the sources are placed on. One of `"v"`, `"h"`.

`selectedpoints`
: Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.

`textfont`
: Sets the font for node labels Named list or `go_sankey_textfont()` object.

`uid`
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision`
: Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable:

true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`valueformat` Sets the value formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. String.

`valuesuffix` Adds a unit to follow the value in the hover tooltip. Add a space if a separation is necessary from the value. String.

`visible` Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`row` `add_sankey()`: Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`go_sankey_domain()`: If there is a layout grid, use the domain for this row in the grid for this sankey trace. Integer in the interval [0, Inf].

`col` Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`skip_invalid` If `TRUE`, silently drop invalid properties instead of raising an error.

`arg` Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

`column` If there is a layout grid, use the domain for this column in the grid for this sankey trace. Integer in the interval [0, Inf].

`x` `go_sankey_domain()`: Sets the horizontal domain of this sankey trace (in plot fraction). Vector or list of 2 element(s).

`go_sankey_node()`: The normalized horizontal position of the node. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`y` `go_sankey_domain()`: Sets the vertical domain of this sankey trace (in plot fraction). Vector or list of 2 element(s).

`go_sankey_node()`: The normalized vertical position of the node. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`align` `go_sankey_hoverlabel()`, `go_sankey_link_hoverlabel()`, `go_sankey_node_hoverlabel()`: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`go_sankey_node()`: Sets the alignment method used to position the nodes along the horizontal axis. One of `"justify"`, `"left"`, `"right"`, `"center"`.

`bgcolor` Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`bordercolor` Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

font
: go_sankey_hoverlabel(): Sets the font used in hover labels. Named list or go_sankey_hoverlabel_font() object.
go_sankey_legendgrouptitle(): Sets this legend group's title font. Named list or go_sankey_legendgrouptitle_font() object.
go_sankey_link_hoverlabel(): Sets the font used in hover labels. Named list or go_sankey_link_hoverlabel_font() object.
go_sankey_node_hoverlabel(): Sets the font used in hover labels. Named list or go_sankey_node_hoverlabel_font() object.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color
: go_sankey_hoverlabel_font(), go_sankey_link_hoverlabel_font(), go_sankey_node_hoverlab
Color: a CSS color string or R color name, or a vector of these.
go_sankey_legendgrouptitle_font(), go_sankey_textfont(): Color: a CSS color string or R color name.
go_sankey_link(): Sets the 'link' color. It can be a single value, or an array for specifying color for each 'link'. If 'link.color' is omitted, then by default, a translucent grey link will be used. Color: a CSS color string or R color name, or a vector of these.
go_sankey_link_line(): Sets the color of the 'line' around each 'link'. Color: a CSS color string or R color name, or a vector of these.
go_sankey_node(): Sets the 'node' color. It can be a single value, or an array for specifying color for each 'node'. If 'node.color' is omitted, then the default 'Plotly' color palette will be cycled through to have a variety of colors. These defaults are not fully opaque, to allow some visibility of what is beneath the node. Color: a CSS color string or R color name, or a vector of these.
go_sankey_node_line(): Sets the color of the 'line' around each 'node'. Color: a CSS color string or R color name, or a vector of these.

family
: go_sankey_hoverlabel_font(), go_sankey_link_hoverlabel_font(), go_sankey_node_hoverlab
HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.
go_sankey_legendgrouptitle_font(), go_sankey_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: go_sankey_hoverlabel_font(), go_sankey_link_hoverlabel_font(), go_sankey_node_hoverlab
Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination

of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_sankey_legendgrouptitle_font(), go_sankey_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow

go_sankey_hoverlabel_font(), go_sankey_link_hoverlabel_font(), go_sankey_node_hoverlab Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_sankey_legendgrouptitle_font(), go_sankey_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size

go_sankey_hoverlabel_font(), go_sankey_link_hoverlabel_font(), go_sankey_node_hoverlab Number in the interval [1, Inf], or a vector of these.

go_sankey_legendgrouptitle_font(), go_sankey_textfont(): Number in the interval [1, Inf].

style

go_sankey_hoverlabel_font(), go_sankey_link_hoverlabel_font(), go_sankey_node_hoverlab Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_sankey_legendgrouptitle_font(), go_sankey_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase

go_sankey_hoverlabel_font(), go_sankey_link_hoverlabel_font(), go_sankey_node_hoverlab Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_sankey_legendgrouptitle_font(), go_sankey_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant

go_sankey_hoverlabel_font(), go_sankey_link_hoverlabel_font(), go_sankey_node_hoverlab Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_sankey_legendgrouptitle_font(), go_sankey_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight

go_sankey_hoverlabel_font(), go_sankey_link_hoverlabel_font(), go_sankey_node_hoverlab Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_sankey_legendgrouptitle_font(), go_sankey_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

text

Sets the title of the legend group. String.

`arrowlen` Sets the length (in px) of the links arrow, if 0 no arrow will be drawn. Number in the interval [0, Inf].

`colorscales` List of named lists or `go_sankey_link_concentrationscales()` objects.

`hovercolor` Sets the 'link' hover color. It can be a single value, or an array for specifying hover colors for each 'link'. If 'link.hovercolor' is omitted, then by default, links will become slightly more opaque when hovered over. Color: a CSS color string or R color name, or a vector of these.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'value' and 'label'. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`label` `go_sankey_link()`: The shown name of the link. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_sankey_link_concentrationscales()`: The label of the links to color based on their concentration within a flow. String.

`go_sankey_node()`: The shown name of the node. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`line` `go_sankey_link()`: Named list or `go_sankey_link_line()` object.

`go_sankey_node()`: Named list or `go_sankey_node_line()` object.

`sort` `go_sankey_link()`: For 'auto' (the default), the order of links attached to each node is determined automatically by the layout. For 'input', the order of links at each node is kept the same as the order in the input array. One of `"auto"`, `"input"`.

go_sankey_node(): For ‘auto‘ (the default), the vertical order of nodes will be determined automatically by the layout. For ‘input‘, the vertical order of nodes is kept the same as the order in the input array. One of "auto", "input".

source — An integer number ‘[0..nodes.length - 1]‘ that represents the source node. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

target — An integer number ‘[0..nodes.length - 1]‘ that represents the target node. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

value — A numeric value representing the flow volume value. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

cmax — Sets the upper bound of the color domain. Number.

cmin — Sets the lower bound of the color domain. Number.

colorscale — Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘cmin‘ and ‘cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

width — go_sankey_link_line(): Sets the width (in px) of the ‘line‘ around each ‘link‘. Number in the interval [0, Inf], or a vector of these.

go_sankey_node_line(): Sets the width (in px) of the ‘line‘ around each ‘node‘. Number in the interval [0, Inf], or a vector of these.

groups — Groups of nodes. Each group is defined by an array with the indices of the nodes it contains. Multiple groups can be specified. Vector or list of up to 1 element(s).

pad — Sets the padding (in px) between the ‘nodes‘. Number in the interval [0, Inf].

thickness — Sets the thickness (in px) of the ‘nodes‘. Number in the interval [1, Inf].

Details

go_sankey() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_sankey() creates an object and adds it to a figure.

Nested objects documented here:

go_sankey_domain() sankey.domain

`go_sankey_hoverlabel()` `sankey.hoverlabel`

`go_sankey_hoverlabel_font()` `sankey.hoverlabel.font`: Sets the font used in hover labels.

`go_sankey_legendgrouptitle()` `sankey.legendgrouptitle`

`go_sankey_legendgrouptitle_font()` `sankey.legendgrouptitle.font`: Sets this legend group's title font.

`go_sankey_link()` `sankey.link`: The links of the Sankey plot.

`go_sankey_link_concentrationscales()` `sankey.link.concentrationscales`

`go_sankey_link_hoverlabel()` `sankey.link.hoverlabel`

`go_sankey_link_hoverlabel_font()` `sankey.link.hoverlabel.font`: Sets the font used in hover labels.

`go_sankey_link_line()` `sankey.link.line`

`go_sankey_node()` `sankey.node`: The nodes of the Sankey plot.

`go_sankey_node_hoverlabel()` `sankey.node.hoverlabel`

`go_sankey_node_hoverlabel_font()` `sankey.node.hoverlabel.font`: Sets the font used in hover labels.

`go_sankey_node_line()` `sankey.node.line`

`go_sankey_textfont()` `sankey.textfont`: Sets the font for node labels

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_sankey()
```

---

add_scatter *Scatter traces*

Description

The scatter trace type encompasses line charts, scatter charts, text charts, and bubble charts. The data visualized as scatter point or lines is set in 'x' and 'y'. Text (appearing either on the chart or on hover only) is via 'text'. Bubble charts are achieved by setting 'marker.size' and/or 'marker.color' to numerical arrays.

Usage

```
add_scatter(fig, ...)

## Default S3 method:
add_scatter(
  fig,
  ...,
  data = NULL,
  alignmentgroup = NULL,
  cliponaxis = NULL,
  connectgaps = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  error_x = NULL,
  error_y = NULL,
  fill = NULL,
  fillcolor = NULL,
  fillgradient = NULL,
  fillpattern = NULL,
  groupnorm = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  offsetgroup = NULL,
  opacity = NULL,
  orientation = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  stackgaps = NULL,
  stackgroup = NULL,
  text = NULL,
  textfont = NULL,
```

```
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  yperiod = NULL,
  yperiod0 = NULL,
  yperiodalignment = NULL,
  zorder = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_scatter(
  arg = NULL,
  ...,
  alignmentgroup = NULL,
  cliponaxis = NULL,
  connectgaps = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  error_x = NULL,
  error_y = NULL,
  fill = NULL,
  fillcolor = NULL,
  fillgradient = NULL,
  fillpattern = NULL,
  groupnorm = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
```

```
    hoveron = NULL,
    hovertemplate = NULL,
    hovertemplatefallback = NULL,
    hovertext = NULL,
    ids = NULL,
    legend = NULL,
    legendgroup = NULL,
    legendgrouptitle = NULL,
    legendrank = NULL,
    legendwidth = NULL,
    line = NULL,
    marker = NULL,
    meta = NULL,
    mode = NULL,
    name = NULL,
    offsetgroup = NULL,
    opacity = NULL,
    orientation = NULL,
    selected = NULL,
    selectedpoints = NULL,
    showlegend = NULL,
    stackgaps = NULL,
    stackgroup = NULL,
    text = NULL,
    textfont = NULL,
    textposition = NULL,
    texttemplate = NULL,
    texttemplatefallback = NULL,
    uid = NULL,
    uirevision = NULL,
    unselected = NULL,
    visible = NULL,
    x = NULL,
    x0 = NULL,
    xaxis = NULL,
    xcalendar = NULL,
    xhoverformat = NULL,
    xperiod = NULL,
    xperiod0 = NULL,
    xperiodalignment = NULL,
    y = NULL,
    y0 = NULL,
    yaxis = NULL,
    ycalendar = NULL,
    yhoverformat = NULL,
    yperiod = NULL,
    yperiod0 = NULL,
    yperiodalignment = NULL,
```

```
  zorder = NULL,
  skip_invalid = FALSE
)

go_scatter_error_x(
  arg = NULL,
  ...,
  array = NULL,
  arrayminus = NULL,
  color = NULL,
  copy_ystyle = NULL,
  symmetric = NULL,
  thickness = NULL,
  traceref = NULL,
  tracerefminus = NULL,
  type = NULL,
  value = NULL,
  valueminus = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatter_error_y(
  arg = NULL,
  ...,
  array = NULL,
  arrayminus = NULL,
  color = NULL,
  symmetric = NULL,
  thickness = NULL,
  traceref = NULL,
  tracerefminus = NULL,
  type = NULL,
  value = NULL,
  valueminus = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatter_fillgradient(
  arg = NULL,
  ...,
  colorscale = NULL,
  start = NULL,
  stop = NULL,
  type = NULL,
```

```
  skip_invalid = FALSE
)

go_scatter_fillpattern(
  arg = NULL,
  ...,
  bgcolor = NULL,
  fgcolor = NULL,
  fgopacity = NULL,
  fillmode = NULL,
  path = NULL,
  shape = NULL,
  size = NULL,
  solidity = NULL,
  skip_invalid = FALSE
)

go_scatter_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_scatter_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatter_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
```

```
  text = NULL,
  skip_invalid = FALSE
)

go_scatter_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatter_line(
  arg = NULL,
  ...,
  backoff = NULL,
  color = NULL,
  dash = NULL,
  shape = NULL,
  simplify = NULL,
  smoothing = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatter_marker(
  arg = NULL,
  ...,
  angle = NULL,
  angleref = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  gradient = NULL,
  line = NULL,
```

```
  maxdisplayed = NULL,
  opacity = NULL,
  reversescale = NULL,
  showscale = NULL,
  size = NULL,
  sizemin = NULL,
  sizemode = NULL,
  sizeref = NULL,
  standoff = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_scatter_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
```

```
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_scatter_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatter_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_scatter_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
```

```
  text = NULL,
  skip_invalid = FALSE
)

go_scatter_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatter_marker_gradient(
  arg = NULL,
  ...,
  color = NULL,
  type = NULL,
  skip_invalid = FALSE
)

go_scatter_marker_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  dash = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatter_selected(
  arg = NULL,
  ...,
  marker = NULL,
```

```
  textfont = NULL,
  skip_invalid = FALSE
)

go_scatter_selected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scatter_selected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

go_scatter_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatter_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scatter_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
```

```
  size = NULL,
  skip_invalid = FALSE
)

go_scatter_unselected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
alignmentgroup	Set several traces linked to the same position axis or matching axes to the same alignmentgroup. This controls whether bars compute their positional range dependently or independently. String.
cliponaxis	Determines whether or not markers and text nodes are clipped about the subplot axes. To show markers and text nodes above axis lines and tick labels, make sure to set ‘xaxis.layer‘ and ‘yaxis.layer‘ to "below traces". Logical.
connectgaps	Determines whether or not gaps (i.e. {nan} or missing values) in the provided data arrays are connected. Logical.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
dx	Sets the x coordinate step. See ‘x0‘ for more info. Number.
dy	Sets the y coordinate step. See ‘y0‘ for more info. Number.
error_x	Named list or go_scatter_error_x() object.
error_y	Named list or go_scatter_error_y() object.
fill	Sets the area to fill with a solid color. Defaults to "none" unless this trace is stacked, then it gets "tonexty" ("tonextx") if ‘orientation‘ is "v" ("h") Use with ‘fillcolor‘ if not "none". "tozerox" and "tozeroy" fill to x=0 and y=0 respectively. "tonextx" and "tonexty" fill between the endpoints of this trace and the endpoints of the trace before it, connecting those endpoints with straight lines (to make a stacked area graph); if there is no trace before it, they behave like "tozerox" and "tozeroy". "toself" connects the endpoints of the trace (or each segment of the trace if it has gaps) into a closed shape. "tonext" fills the space between two

traces if one completely encloses the other (eg consecutive contour lines), and behaves like "toself" if there is no trace before it. "tonext" should not be used if one trace does not enclose the other. Traces in a ‘stackgroup‘ will only fill to (or be filled to) other traces in the same group. With multiple ‘stackgroup‘s or some traces stacked and some not, if fill-linked traces are not already consecutive, the later ones will be pushed down in the drawing order. One of `"none"`, `"tozeroy"`, `"tozerox"`, `"tonexty"`, `"tonextx"`, `"toself"`, `"tonext"`.

`fillcolor` Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. If fillgradient is specified, fillcolor is ignored except for setting the background color of the hover label, if any. Color: a CSS color string or R color name.

`fillgradient` Sets a fill gradient. If not specified, the fillcolor is used instead. Named list or `go_scatter_fillgradient()` object.

`fillpattern` Sets the pattern within the marker. Named list or `go_scatter_fillpattern()` object.

`groupnorm` Only relevant when ‘stackgroup‘ is used, and only the first ‘groupnorm‘ found in the ‘stackgroup‘ will be used - including if ‘visible‘ is "legendonly" but not if it is ‘false‘. Sets the normalization for the sum of this ‘stackgroup‘. With "fraction", the value of each trace at each location is divided by the sum of all trace values at that location. "percent" is the same but multiplied by 100 to show percentages. If there are multiple subplots, or multiple ‘stackgroup‘s on one subplot, each will be normalized within its own set. One of `""`, `"fraction"`, `"percent"`.

`hoverinfo` Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_scatter_hoverlabel()` object.

`hoveron` Do the hover effects highlight individual points (markers or line points) or do they highlight filled regions? If the fill is "toself" or "tonext" and there are no markers or text, then the default is "fills", otherwise it is "points". Any combination of `"points"`, `"fills"` joined with `"+"`.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables

with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_scatter_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line`
: `go_scatter()`: Named list or `go_scatter_line()` object.
`go_scatter_marker()`: Named list or `go_scatter_marker_line()` object.

`marker`
: `go_scatter()`: Named list or `go_scatter_marker()` object.
`go_scatter_selected()`: Named list or `go_scatter_selected_marker()` object.
`go_scatter_unselected()`: Named list or `go_scatter_unselected_marker()` object.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders'

'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

`mode` Determines the drawing mode for this scatter trace. If the provided 'mode' includes "text" then the 'text' elements appear at the coordinates. Otherwise, the 'text' elements appear on hover. If there are less than 20 points and the trace is not stacked then the default is "lines+markers". Otherwise, "lines". Any combination of `"lines"`, `"markers"`, `"text"` joined with `"+"`, or one of `"none"`.

`name` `go_scatter()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_scatter_marker_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

`offsetgroup` Set several traces linked to the same position axis or matching axes to the same offsetgroup where bars of the same position coordinate will line up. String.

`opacity` `go_scatter()`: Sets the opacity of the trace. Number in the interval [0, 1].

`go_scatter_marker()`: Sets the marker opacity. Number in the interval [0, 1], or a vector of these.

`go_scatter_selected_marker()`: Sets the marker opacity of selected points. Number in the interval [0, 1].

`go_scatter_unselected_marker()`: Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

`orientation` `go_scatter()`: Only relevant in the following cases: 1. when 'scattermode' is set to "group". 2. when 'stackgroup' is used, and only the first 'orientation' found in the 'stackgroup' will be used - including if 'visible' is "legendonly" but not if it is 'false'. Sets the stacking direction. With "v" ("h"), the y (x) values of subsequent traces are added. Also affects the default value of 'fill'. One of `"v"`, `"h"`.

`go_scatter_marker_colorbar()`: Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`selected` Named list or `go_scatter_selected()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the 'unselected' are turned on for all points, whereas, any other non-array values means no selection all where the 'selected' and 'unselected' styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`stackgaps` Only relevant when 'stackgroup' is used, and only the first 'stackgaps' found in the 'stackgroup' will be used - including if 'visible' is "legendonly" but not if

it is ‘false‘. Determines how we handle locations at which other traces in this group have data but this one does not. With "infer zero" we insert a zero at these locations. With "interpolate" we linearly interpolate between existing values, and extrapolate a constant beyond the existing values. One of `"infer zero"`, `"interpolate"`.

`stackgroup` Set several scatter traces (on the same subplot) to the same stackgroup in order to add their y values (or their x values if ‘orientation‘ is "h"). If blank or omitted this trace will not be stacked. Stacking also turns ‘fill‘ on by default, using "tonexty" ("tonextx") if ‘orientation‘ is "h" ("v") and sets the default ‘mode‘ to "lines" irrespective of point count. You can only stack on a numeric (linear or log) axis. Traces in a ‘stackgroup‘ will only fill to (or be filled to) other traces in the same group. With multiple ‘stackgroup‘s or some traces stacked and some not, if fill-linked traces are not already consecutive, the later ones will be pushed down in the drawing order. String.

`text` `go_scatter()`: Sets text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_scatter_legendgrouptitle()`: Sets the title of the legend group. String.

`go_scatter_marker_colorbar_title()`: Sets the title of the color bar. String.

`textfont` `go_scatter()`: Sets the text font. Named list or `go_scatter_textfont()` object.

`go_scatter_selected()`: Named list or `go_scatter_selected_textfont()` object.

`go_scatter_unselected()`: Named list or `go_scatter_unselected_textfont()` object.

`textposition` Sets the positions of the ‘text‘ elements with respects to the (x,y) coordinates. One of `"top left"`, `"top center"`, `"top right"`, `"middle left"`, `"middle center"`, `"middle right"`, `"bottom left"`, `"bottom center"`, `"bottom right"`, or a vector of these.

`texttemplate` Template string used for rendering the information text that appears on points. Note that this will override ‘textinfo‘. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. String, or a vector of these.

`texttemplatefallback`
Fallback string that's displayed when a variable referenced in a template is miss-

ing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`unselected` Named list or `go_scatter_unselected()` object.

`visible` `go_scatter()`: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`go_scatter_error_x()`, `go_scatter_error_y()`: Determines whether or not this set of error bars is visible. Logical.

`x` `go_scatter()`: Sets the x coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_scatter_marker_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`x0` Alternate to ‘x‘. Builds a linear space of x coordinates. Use with ‘dx‘ where ‘x0‘ is the starting coordinate and ‘dx‘ the step. Any value.

`xaxis` Sets a reference between this trace's x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

`xcalendar` Sets the calendar system to use with ‘x‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

`xhoverformat` Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

xperiod Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the x axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

xperiod0 Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the x0 axis. When ‘x0period‘ is round number of weeks, the ‘x0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

xperiodalignment
Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the x axis. One of "start", "middle", "end".

y go_scatter(): Sets the y coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_scatter_marker_colorbar(): Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

y0 Alternate to ‘y‘. Builds a linear space of y coordinates. Use with ‘dy‘ where ‘y0‘ is the starting coordinate and ‘dy‘ the step. Any value.

yaxis Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: "y", "y2", "y3", ...

ycalendar Sets the calendar system to use with ‘y‘ date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

yhoverformat Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

yperiod Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the y axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

yperiod0 Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the y0 axis. When ‘y0period‘ is round number of weeks, the ‘y0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

yperiodalignment
Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the y axis. One of "start", "middle", "end".

zorder	Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher ‘zorder‘ appear in front of those with lower ‘zorder‘. Integer.
row	Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
secondary_y	If TRUE, use the secondary y axis of the subplot.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
array	Sets the data corresponding the length of each error bar. Values are plotted relative to the underlying data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
arrayminus	Sets the data corresponding the length of each error bar in the bottom (left) direction for vertical (horizontal) bars Values are plotted relative to the underlying data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
color	go_scatter_error_x(), go_scatter_error_y(): Sets the stroke color of the error bars. Color: a CSS color string or R color name. go_scatter_hoverlabel_font(), go_scatter_textfont(): Color: a CSS color string or R color name, or a vector of these. go_scatter_legendgrouptitle_font(), go_scatter_marker_colorbar_tickfont(), go_scatter_marker_colorbar_title_font(): Color: a CSS color string or R color name. go_scatter_line(): Sets the line color. Color: a CSS color string or R color name. go_scatter_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.cmin‘ and ‘marker.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scatter.marker.colorscale, or a vector of these. go_scatter_marker_gradient(): Sets the final color of the gradient fill: the center color for radial, the right for horizontal, or the bottom for vertical. Color: a CSS color string or R color name, or a vector of these. go_scatter_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scatter.marker.line.colorscale, or a vector of these. go_scatter_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name. go_scatter_selected_textfont(): Sets the text font color of selected points. Color: a CSS color string or R color name.

go_scatter_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

go_scatter_unselected_textfont(): Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

copy_ystyle Logical.

symmetric Determines whether or not the error bars have the same length in both direction (top/bottom for vertical bars, left/right for horizontal bars. Logical.

thickness go_scatter_error_x(), go_scatter_error_y(): Sets the thickness (in px) of the error bars. Number in the interval [0, Inf].

go_scatter_marker_colorbar(): Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

traceref Integer in the interval [0, Inf].

tracerefminus Integer in the interval [0, Inf].

type go_scatter_error_x(),go_scatter_error_y(): Determines the rule used to generate the error bars. If "constant", the bar lengths are of a constant value. Set this constant in 'value'. If "percent", the bar lengths correspond to a percentage of underlying data. Set this percentage in 'value'. If "sqrt", the bar lengths correspond to the square of the underlying data. If "data", the bar lengths are set with data set 'array'. One of "percent", "constant", "sqrt", "data".

go_scatter_fillgradient(): Sets the type/orientation of the color gradient for the fill. Defaults to "none". One of "radial", "horizontal", "vertical", "none".

go_scatter_marker_gradient(): Sets the type of gradient used to fill the markers One of "radial", "horizontal", "vertical", "none", or a vector of these.

value go_scatter_error_x(), go_scatter_error_y(): Sets the value of either the percentage (if 'type' is set to "percent") or the constant (if 'type' is set to "constant") corresponding to the lengths of the error bars. Number in the interval [0, Inf].

go_scatter_marker_colorbar_tickformatstop(): string - dtickformat for described zoom level, the same as "tickformat" String.

valueminus Sets the value of either the percentage (if 'type' is set to "percent") or the constant (if 'type' is set to "constant") corresponding to the lengths of the error bars in the bottom (left) direction for vertical (horizontal) bars Number in the interval [0, Inf].

width go_scatter_error_x(), go_scatter_error_y(): Sets the width (in px) of the cross-bar at both ends of the error bars. Number in the interval [0, Inf].

go_scatter_line(): Sets the line width (in px). Number in the interval [0, Inf].

go_scatter_marker_line(): Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

`colorscale` `go_scatter_fillgradient()`: Sets the fill gradient colors as a color scale. The color scale is interpreted as a gradient applied in the direction specified by "orientation", from the lowest to the highest value of the scatter plot along that axis, or from the center to the most distant point from it, if orientation is "radial". Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_scatter_marker()`: Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Re
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_scatter_marker_line()`: Sets the colorscale. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.line.cmin‘ and ‘marker.line.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,H
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`start` Sets the gradient start value. It is given as the absolute position on the axis determined by the orientiation. E.g., if orientation is "horizontal", the gradient will be horizontal and start from the x-position given by start. If omitted, the gradient starts at the lowest value of the trace along the respective axis. Ignored if orientation is "radial". Number.

`stop` Sets the gradient end value. It is given as the absolute position on the axis determined by the orientiation. E.g., if orientation is "horizontal", the gradient will be horizontal and end at the x-position given by end. If omitted, the gradient ends at the highest value of the trace along the respective axis. Ignored if orientation is "radial". Number.

`bgcolor` `go_scatter_fillpattern()`: When there is no colorscale sets the color of background pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "overlay". Otherwise, defaults to a transparent background. Color: a CSS color string or R color name, or a vector of these.

`go_scatter_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`go_scatter_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

fgcolor — When there is no colorscale sets the color of foreground pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "replace". Otherwise, defaults to dark grey or white to increase contrast with the ‘bgcolor‘. Color: a CSS color string or R color name, or a vector of these.

fgopacity — Sets the opacity of the foreground pattern fill. Defaults to a 0.5 when ‘fillmode‘ is "overlay". Otherwise, defaults to 1. Number in the interval [0, 1].

fillmode — Determines whether ‘marker.color‘ should be used as a default to ‘bgcolor‘ or a ‘fgcolor‘. One of `"replace"`, `"overlay"`.

path — Sets a custom path for pattern fill. Use with no ‘shape‘ or ‘solidity‘, provide an SVG path string for the regions of the square from (0,0) to (‘size‘,‘size‘) to color. String, or a vector of these.

shape — `go_scatter_fillpattern()`: Sets the shape of the pattern fill. By default, no pattern is used for filling the area. One of `""`, `"/"`, `"\\"`, `"x"`, `"-"`, `"|"`, `"+"`, `"."`, or a vector of these.

`go_scatter_line()`: Determines the line shape. With "spline" the lines are drawn using spline interpolation. The other available values correspond to step-wise line shapes. One of `"linear"`, `"spline"`, `"hv"`, `"vh"`, `"hvh"`, `"vhv"`.

size — `go_scatter_fillpattern()`: Sets the size of unit squares of the pattern fill in pixels, which corresponds to the interval of repetition of the pattern. Number in the interval [0, Inf], or a vector of these.

`go_scatter_hoverlabel_font()`, `go_scatter_textfont()`: Number in the interval [1, Inf], or a vector of these.

`go_scatter_legendgrouptitle_font()`, `go_scatter_marker_colorbar_tickfont()`, `go_scatter_marker_colorbar_title_font()`: Number in the interval [1, Inf].

`go_scatter_marker()`: Sets the marker size (in px). Number in the interval [0, Inf], or a vector of these.

`go_scatter_selected_marker()`: Sets the marker size of selected points. Number in the interval [0, Inf].

`go_scatter_unselected_marker()`: Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

solidity — Sets the solidity of the pattern fill. Solidity is roughly the fraction of the area filled by the pattern. Solidity of 0 shows only the background color without pattern and solidty of 1 shows only the foreground color without pattern. Number in the interval [0, 1], or a vector of these.

align — Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

bordercolor — `go_scatter_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`go_scatter_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

font — `go_scatter_hoverlabel()`: Sets the font used in hover labels. Named list or `go_scatter_hoverlabel_font()` object.

`go_scatter_legendgrouptitle()`: Sets this legend group’s title font. Named list or `go_scatter_legendgrouptitle_font()` object.

go_scatter_marker_colorbar_title(): Sets this color bar's title font. Named list or go_scatter_marker_colorbar_title_font() object.

namelength — Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow — Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

family — go_scatter_hoverlabel_font(), go_scatter_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

go_scatter_legendgrouptitle_font(), go_scatter_marker_colorbar_tickfont(), go_scatter_marker_colorbar_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition — go_scatter_hoverlabel_font(), go_scatter_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_scatter_legendgrouptitle_font(), go_scatter_marker_colorbar_tickfont(), go_scatter_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow — go_scatter_hoverlabel_font(), go_scatter_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_scatter_legendgrouptitle_font(), go_scatter_marker_colorbar_tickfont(), go_scatter_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

style — go_scatter_hoverlabel_font(), go_scatter_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_scatter_legendgrouptitle_font(), go_scatter_marker_colorbar_tickfont(), go_scatter_marker_colorbar_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_scatter_hoverlabel_font(), go_scatter_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.
go_scatter_legendgrouptitle_font(), go_scatter_marker_colorbar_tickfont(), go_scatter_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_scatter_hoverlabel_font(), go_scatter_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.
go_scatter_legendgrouptitle_font(), go_scatter_marker_colorbar_tickfont(), go_scatter_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_scatter_hoverlabel_font(), go_scatter_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.
go_scatter_legendgrouptitle_font(), go_scatter_marker_colorbar_tickfont(), go_scatter_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

backoff
: Sets the line back off from the end point of the nth line segment (in px). This option is useful e.g. to avoid overlap with arrowhead markers. With "auto" the lines would trim before markers if ‘marker.angleref‘ is set to "previous". Number in the interval [0, Inf], or a vector of these.

dash
: go_scatter_line(): Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\
go_scatter_marker_line(): Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\
or a vector of these.

simplify
: Simplifies lines by removing nearly-collinear points. When transitioning lines, it may be desirable to disable this so that the number of points along the resulting SVG path is unaffected. Logical.

smoothing
: Has an effect only if ‘shape‘ is set to "spline" Sets the amount of smoothing. 0 corresponds to no smoothing (equivalent to a "linear" shape). Number in the interval [0, 1.3].

angle
: Sets the marker angle in respect to ‘angleref‘. Angle in degrees, or a vector of these.

angleref
: Sets the reference for marker angle. With "previous", angle 0 points along the line from the previous point to this one. With "up", angle 0 points toward the top of the screen. One of "previous", "up".

autocolorscale
: go_scatter_marker(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an

effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`go_scatter_marker_line()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`cauto` `go_scatter_marker()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

`go_scatter_marker_line()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

`cmax` `go_scatter_marker()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

`go_scatter_marker_line()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

`cmid` `go_scatter_marker()`: Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

`go_scatter_marker_line()`: Sets the mid-point of the color domain by scaling ‘marker.line.cmin‘ and/or ‘marker.line.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

`cmin` `go_scatter_marker()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`go_scatter_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes

are set in the layout, under 'layout.coloraxis', 'layout.coloraxis2', etc. Note that multiple color scales can be linked to the same color axis. Subplot id: "coloraxis", "coloraxis2", "coloraxis3", ...

`colorbar` Named list or `go_scatter_marker_colorbar()` object.

`gradient` Named list or `go_scatter_marker_gradient()` object.

`maxdisplayed` Sets a maximum number of points to be drawn on the graph. 0 corresponds to no limit. Number in the interval [0, Inf].

`reversescale` `go_scatter_marker()`: Reverses the color mapping if true. Has an effect only if in 'marker.color' is set to a numerical array. If true, 'marker.cmin' will correspond to the last color in the array and 'marker.cmax' will correspond to the first color. Logical.

`go_scatter_marker_line()`: Reverses the color mapping if true. Has an effect only if in 'marker.line.color' is set to a numerical array. If true, 'marker.line.cmin' will correspond to the last color in the array and 'marker.line.cmax' will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if in 'marker.color' is set to a numerical array. Logical.

`sizemin` Has an effect only if 'marker.size' is set to a numerical array. Sets the minimum size (in px) of the rendered marker points. Number in the interval [0, Inf].

`sizemode` Has an effect only if 'marker.size' is set to a numerical array. Sets the rule for which the data in 'size' is converted to pixels. One of "diameter", "area".

`sizeref` Has an effect only if 'marker.size' is set to a numerical array. Sets the scale factor used to determine the rendered size of marker points. Use with 'sizemin' and 'sizemode'. Number.

`standoff` Moves the marker away from the data point in the direction of 'angle' (in px). This can be useful for example if you have another marker at this location and you want to point an arrowhead marker at it. Number in the interval [0, Inf], or a vector of these.

`symbol` Sets the marker symbol type. Adding 100 is equivalent to appending "-open" to a symbol name. Adding 200 is equivalent to appending "-dot" to a symbol name. Adding 300 is equivalent to appending "-open-dot" or "dot-open" to a symbol name. One of "0", "circle", "100", "circle-open", "200", "circle-dot", "300", "circle-open-dot", "1", "square", "101", "square-open", "201", "square-dot", "301", "square-open-dot", "2", "diamond", "102", "diamond-open", "202", "diamond-dot", "302", "diamond-open-dot", "3", "cross", "103", "cross-open", "203", "cross-dot", "303", "cross-open-dot", "4", "x", "104", "x-open", "204", "x-dot", "304", "x-open-dot", "5", "triangle-up", "105", "triangle-up-open", "205", "triangle-up-dot", "305", "triangle-up-open-dot", "6", "triangle-down", "106", "triangle-down-open", "206", "triangle-down-dot", "306", "triangle-down-open-dot", "7", "triangle-left", "107", "triangle-left-open", "207", "triangle-left-dot", "307", "triangle-left-open-dot", "8", "triangle-right", "108", "triangle-right-open", "208", "triangle-right-dot", "308", "triangle-right-open-do "9", "triangle-ne", "109", "triangle-ne-open", "209", "triangle-ne-dot", "309", "triangle-ne-open-dot", "10", "triangle-se", "110", "triangle-se-open", "210", "triangle-se-dot", "310", "triangle-se-open-dot", "11", "triangle-sw",

"111", "triangle-sw-open", "211", "triangle-sw-dot", "311", "triangle-sw-open-dot", "12", "triangle-nw", "112", "triangle-nw-open", "212", "triangle-nw-dot", "312", "triangle-nw-open-dot", "13", "pentagon", "113", "pentagon-open", "213", "pentagon-dot", "313", "pentagon-open-dot", "14", "hexagon", "114", "hexagon-open", "214", "hexagon-dot", "314", "hexagon-open-dot", "15", "hexagon2", "115", "hexagon2-open", "215", "hexagon2-dot", "315", "hexagon2-open-dot", "16", "octagon", "116", "octagon-open", "216", "octagon-dot", "316", "octagon-open-dot", "17", "star", "117", "star-open", "217", "star-dot", "317", "star-open-dot", "18", "hexagram", "118", "hexagram-open", "218", "hexagram-dot", "318", "hexagram-open-dot", "19", "star-triangle-up", "119", "star-triangle-up-open", "219", "star-triangle-up-dot", "319", "star-triangle-up-open-dot", "20", "star-triangle-down", "120", "star-triangle-down-open "220", "star-triangle-down-dot", "320", "star-triangle-down-open-dot", "21", "star-square", "121", "star-square-open", "221", "star-square-dot", "321", "star-square-open-dot", "22", "star-diamond", "122", "star-diamond-open", "222", "star-diamond-dot", "322", "star-diamond-open-dot", "23", "diamond-tall", "123", "diamond-tall-open", "223", "diamond-tall-dot", "323", "diamond-tall-open-dot", "24", "diamond-wide", "124", "diamond-wide-open", "224", "diamond-wide-dot", "324", "diamond-wide-open-dot", "25", "hourglass", "125", "hourglass-open", "26", "bowtie", "126", "bowtie-open", "27", "circle-cross", "127", "circle-cross-open", "28", "circle-x", "128", "circle-x-open", "29", "square-cross", "129", "square-cross-open", "30", "square-x", "130", "square-x-open", "31", "diamond-cross", "131", "diamond-cross-open", "32", "diamond-x", "132", "diamond-x-open", "33", "cross-thin", "133", "cross-thin-open", "34", "x-thin", "134", "x-thin-open", "35", "asterisk", "135", "asterisk-open", "36", "hash", "136", "hash-open", "236", "hash-dot", "336", "hash-open-dot", "37", "y-up", "137", "y-up-open", "38", "y-down", "138", "y-down-open", "39", "y-left", "139", "y-left-open", "40", "y-right", "140", "y-right-open", "41", "line-ew", "141", "line-ew-open", "42", "line-ns", "142", "line-ns-open", "43", "line-ne", "143", "line-ne-open", "44", "line-nw", "144", "line-nw-open", "45", "arrow-up", "145", "arrow-up-open", "46", "arrow-down", "146", "arrow-down-open", "47", "arrow-left", "147", "arrow-left-open", "48", "arrow-right", "148", "arrow-right-open", "49", "arrow-bar-up", "149", "arrow-bar-up-open", "50", "arrow-bar-down", "150", "arrow-bar-down-open", "51", "arrow-bar-left", "151", "arrow-bar-left-open", "52", "arrow-bar-right", "152", "arrow-bar-right-open", "53", "arrow", "153", "arrow-open", "54", "arrow-wide", "154", "arrow-wide-open", 0, 100, 200, 300, 1, 101, 201, 301, 2, 102, 202, 302, 3, 103, 203, 303, 4, 104, 204, 304, 5, 105, 205, 305, 6, 106, 206, 306, 7, 107, 207, 307, 8, 108, 208, 308, 9, 109, 209, 309, 10, 110, 210, 310, 11, 111, 211, 311, 12, 112, 212, 312, 13, 113, 213, 313, 14, 114, 214, 314, 15, 115, 215, 315, 16, 116, 216, 316, 17, 117, 217, 317, 18, 118, 218, 318, 19, 119, 219, 319, 20, 120, 220, 320, 21, 121, 221, 321, 22, 122, 222, 322, 23, 123, 223, 323, 24, 124, 224, 324, 25, 125, 26, 126, 27, 127, 28, 128, 29, 129, 30, 130, 31, 131, 32, 132, 33, 133, 34, 134, 35, 135, 36, 136, 236, 336, 37, 137, 38, 138, 39, 139, 40, 140, 41, 141, 42, 142, 43, 143, 44, 144, 45, 145, 46, 146, 47, 147, 48, 148, 49, 149, 50, 150, 51, 151, 52, 152, 53, 153, 54, 154, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

dtick Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

exponentformat Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

labelalias Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

len Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

lenmode Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of "fraction", "pixels".

minexponent Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

nticks Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

outlinecolor Sets the axis line color. Color: a CSS color string or R color name.

outlinewidth Sets the width (in px) of the axis line. Number in the interval [0, Inf].

separatethousands
If "true", even 4-digit integers are separated Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".

showticksuffix Same as ‘showtickprefix‘ but for tick suffixes. One of "all", "first", "last", "none".

thicknessmode Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of "fraction", "pixels".

tick0 Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar's tick label font Named list or go_scatter_marker_colorbar_tickfont() object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or go_scatter_marker_colorbar_tickformatstop() objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelstep Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

ticklen Sets the tick length (in px). Number in the interval [0, Inf].

tickmode Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are

provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_scatter_marker_colorbar_title()` object.

`xanchor` Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`yanchor` Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

`ypad` Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

`yref` Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

`dtickrange` range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

`enabled` Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

`templateitemname`
Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching

its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`side` Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.

Details

`go_scatter()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_scatter()` creates an object and adds it to a figure.

Nested objects documented here:

`go_scatter_error_x()` `scatter.error_x`

`go_scatter_error_y()` `scatter.error_y`

`go_scatter_fillgradient()` `scatter.fillgradient`: Sets a fill gradient. If not specified, the fillcolor is used instead.

`go_scatter_fillpattern()` `scatter.fillpattern`: Sets the pattern within the marker.

`go_scatter_hoverlabel()` `scatter.hoverlabel`

`go_scatter_hoverlabel_font()` `scatter.hoverlabel.font`: Sets the font used in hover labels.

`go_scatter_legendgrouptitle()` `scatter.legendgrouptitle`

`go_scatter_legendgrouptitle_font()` `scatter.legendgrouptitle.font`: Sets this legend group's title font.

`go_scatter_line()` `scatter.line`

`go_scatter_marker()` `scatter.marker`

`go_scatter_marker_colorbar()` `scatter.marker.colorbar`

`go_scatter_marker_colorbar_tickfont()` `scatter.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_scatter_marker_colorbar_tickformatstop()` `scatter.marker.colorbar.tickformatstop`

`go_scatter_marker_colorbar_title()` `scatter.marker.colorbar.title`

`go_scatter_marker_colorbar_title_font()` `scatter.marker.colorbar.title.font`: Sets this color bar's title font.

`go_scatter_marker_gradient()` `scatter.marker.gradient`

`go_scatter_marker_line()` `scatter.marker.line`

`go_scatter_selected()` `scatter.selected`

`go_scatter_selected_marker()` `scatter.selected.marker`

`go_scatter_selected_textfont()` `scatter.selected.textfont`

`go_scatter_textfont()` `scatter.textfont`: Sets the text font.

`go_scatter_unselected()` `scatter.unselected`

`go_scatter_unselected_marker()` `scatter.unselected.marker`

`go_scatter_unselected_textfont()` `scatter.unselected.textfont`

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_scatter(x = 1:3, y = c(2, 1, 3), mode = "markers", marker_color = "red")
```

---

add_scatter3d *Scatter3d traces*

Description

The data visualized as scatter point or lines in 3D dimension is set in 'x', 'y', 'z'. Text (appearing either on the chart or on hover only) is via 'text'. Bubble charts are achieved by setting 'marker.size' and/or 'marker.color' Projections are achieved via 'projection'. Surface fills are achieved via 'surfaceaxis'.

Usage

```
add_scatter3d(fig, ...)

## Default S3 method:
add_scatter3d(
  fig,
  ...,
  data = NULL,
  connectgaps = NULL,
  customdata = NULL,
  error_x = NULL,
  error_y = NULL,
  error_z = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
```

```
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  projection = NULL,
  scene = NULL,
  showlegend = NULL,
  surfaceaxis = NULL,
  surfacecolor = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  y = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  z = NULL,
  zcalendar = NULL,
  zhoverformat = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_scatter3d(
  arg = NULL,
  ...,
  connectgaps = NULL,
  customdata = NULL,
  error_x = NULL,
  error_y = NULL,
  error_z = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
```

```
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  projection = NULL,
  scene = NULL,
  showlegend = NULL,
  surfaceaxis = NULL,
  surfacecolor = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  y = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  z = NULL,
  zcalendar = NULL,
  zhoverformat = NULL,
  skip_invalid = FALSE
)

go_scatter3d_error_x(
  arg = NULL,
  ...,
  array = NULL,
  arrayminus = NULL,
  color = NULL,
  copy_zstyle = NULL,
  symmetric = NULL,
  thickness = NULL,
  traceref = NULL,
  tracerefminus = NULL,
  type = NULL,
  value = NULL,
```

```
  valueminus = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatter3d_error_y(
  arg = NULL,
  ...,
  array = NULL,
  arrayminus = NULL,
  color = NULL,
  copy_zstyle = NULL,
  symmetric = NULL,
  thickness = NULL,
  traceref = NULL,
  tracerefminus = NULL,
  type = NULL,
  value = NULL,
  valueminus = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatter3d_error_z(
  arg = NULL,
  ...,
  array = NULL,
  arrayminus = NULL,
  color = NULL,
  symmetric = NULL,
  thickness = NULL,
  traceref = NULL,
  tracerefminus = NULL,
  type = NULL,
  value = NULL,
  valueminus = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatter3d_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
```

```
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_scatter3d_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatter3d_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scatter3d_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatter3d_line(
  arg = NULL,
  ...,
```

```
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  dash = NULL,
  reversescale = NULL,
  showscale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatter3d_line_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
```

```
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_scatter3d_line_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatter3d_line_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_scatter3d_line_colorbar_title(
  arg = NULL,
```

```
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scatter3d_line_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatter3d_marker(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  line = NULL,
  opacity = NULL,
  reversescale = NULL,
  showscale = NULL,
  size = NULL,
  sizemin = NULL,
  sizemode = NULL,
  sizeref = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_scatter3d_marker_colorbar(
  arg = NULL,
```

```
...,
bgcolor = NULL,
bordercolor = NULL,
borderwidth = NULL,
dtick = NULL,
exponentformat = NULL,
labelalias = NULL,
len = NULL,
lenmode = NULL,
minexponent = NULL,
nticks = NULL,
orientation = NULL,
outlinecolor = NULL,
outlinewidth = NULL,
separatethousands = NULL,
showexponent = NULL,
showticklabels = NULL,
showtickprefix = NULL,
showticksuffix = NULL,
thickness = NULL,
thicknessmode = NULL,
tick0 = NULL,
tickangle = NULL,
tickcolor = NULL,
tickfont = NULL,
tickformat = NULL,
tickformatstops = NULL,
ticklabeloverflow = NULL,
ticklabelposition = NULL,
ticklabelstep = NULL,
ticklen = NULL,
tickmode = NULL,
tickprefix = NULL,
ticks = NULL,
ticksuffix = NULL,
ticktext = NULL,
tickvals = NULL,
tickwidth = NULL,
title = NULL,
x = NULL,
xanchor = NULL,
xpad = NULL,
xref = NULL,
y = NULL,
yanchor = NULL,
ypad = NULL,
yref = NULL,
skip_invalid = FALSE
```

```
)

go_scatter3d_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatter3d_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_scatter3d_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scatter3d_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
```

```
  weight = NULL,
  skip_invalid = FALSE
)

go_scatter3d_marker_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatter3d_projection(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_scatter3d_projection_x(
  arg = NULL,
  ...,
  opacity = NULL,
  scale = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_scatter3d_projection_y(
  arg = NULL,
  ...,
  opacity = NULL,
  scale = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_scatter3d_projection_z(
```

```
  arg = NULL,
  ...,
  opacity = NULL,
  scale = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_scatter3d_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  size = NULL,
  style = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
connectgaps	Determines whether or not gaps (i.e. {nan} or missing values) in the provided data arrays are connected. Logical.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
error_x	Named list or go_scatter3d_error_x() object.
error_y	Named list or go_scatter3d_error_y() object.
error_z	Named list or go_scatter3d_error_z() object.
hoverinfo	Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of "x", "y", "z", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_scatter3d_hoverlabel() object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_},

{%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets text elements associated with each (x,y,z) triplet. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y,z) coordinates. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_scatter3d_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

line
: go_scatter3d(): Named list or go_scatter3d_line() object.
go_scatter3d_marker(): Named list or go_scatter3d_marker_line() object.

marker
: Named list or go_scatter3d_marker() object.

meta
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

mode
: Determines the drawing mode for this scatter trace. If the provided ‘mode‘ includes "text" then the ‘text‘ elements appear at the coordinates. Otherwise, the ‘text‘ elements appear on hover. If there are less than 20 points and the trace is not stacked then the default is "lines+markers". Otherwise, "lines". Any combination of "lines", "markers", "text" joined with "+", or one of "none".

name
: go_scatter3d(): Sets the trace name. The trace name appears as the legend item and on hover. String.
go_scatter3d_line_colorbar_tickformatstop(),go_scatter3d_marker_colorbar_tickformats
When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

opacity
: go_scatter3d(): Sets the opacity of the trace. Number in the interval [0, 1].
go_scatter3d_marker(): Sets the marker opacity. Note that the marker opacity for scatter3d traces must be a scalar value for performance reasons. To set a blending opacity value (i.e. which is not transparent), set "marker.color" to an rgba color and use its alpha channel. Number in the interval [0, 1].
go_scatter3d_projection_x(),go_scatter3d_projection_y(),go_scatter3d_projection_z():
Sets the projection color. Number in the interval [0, 1].

projection
: Named list or go_scatter3d_projection() object.

scene
: Sets a reference between this trace's 3D coordinate system and a 3D scene. If "scene" (the default value), the (x,y,z) coordinates refer to ‘layout.scene‘. If "scene2", the (x,y,z) coordinates refer to ‘layout.scene2‘, and so on. Subplot id: "scene", "scene2", "scene3", ...

showlegend
: Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

surfaceaxis
: If "-1", the scatter points are not fill with a surface If 0, 1, 2, the scatter points are filled with a Delaunay surface about the x, y, z respectively. One of -1, 0, 1, 2.

surfacecolor
: Sets the surface fill color. Color: a CSS color string or R color name.

text
: go_scatter3d(): Sets text elements associated with each (x,y,z) triplet. If a single string, the same string appears over all the data points. If an array of

string, the items are mapped in order to the this trace's (x,y,z) coordinates. If trace 'hoverinfo' contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_scatter3d_legendgrouptitle()`: Sets the title of the legend group. String.

`go_scatter3d_line_colorbar_title()`, `go_scatter3d_marker_colorbar_title()`: Sets the title of the color bar. String.

`textfont` Sets the text font. Named list or `go_scatter3d_textfont()` object.

`textposition` Sets the positions of the 'text' elements with respects to the (x,y) coordinates. One of `"top left"`, `"top center"`, `"top right"`, `"middle left"`, `"middle center"`, `"middle right"`, `"bottom left"`, `"bottom center"`, `"bottom right"`, or a vector of these.

`texttemplate` Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. String, or a vector of these.

`texttemplatefallback`

Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

`visible` `go_scatter3d()`: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`go_scatter3d_error_x()`, `go_scatter3d_error_y()`, `go_scatter3d_error_z()`: Determines whether or not this set of error bars is visible. Logical.

x
: `go_scatter3d()`: Sets the x coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  `go_scatter3d_line_colorbar()`, `go_scatter3d_marker_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

  `go_scatter3d_projection()`: Named list or `go_scatter3d_projection_x()` object.

xcalendar
: Sets the calendar system to use with ‘x‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

xhoverformat
: Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

y
: `go_scatter3d()`: Sets the y coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  `go_scatter3d_line_colorbar()`, `go_scatter3d_marker_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

  `go_scatter3d_projection()`: Named list or `go_scatter3d_projection_y()` object.

ycalendar
: Sets the calendar system to use with ‘y‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

yhoverformat
: Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

z
: `go_scatter3d()`: Sets the z coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_scatter3d_projection(): Named list or go_scatter3d_projection_z() object.

zcalendar — Sets the calendar system to use with ‘z‘ date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

zhoverformat — Sets the hover text formatting rule for ‘z‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘zaxis.hoverformat‘. String.

row — Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.

col — Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

skip_invalid — If TRUE, silently drop invalid properties instead of raising an error.

arg — Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

array — Sets the data corresponding the length of each error bar. Values are plotted relative to the underlying data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

arrayminus — Sets the data corresponding the length of each error bar in the bottom (left) direction for vertical (horizontal) bars Values are plotted relative to the underlying data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

color — go_scatter3d_error_x(), go_scatter3d_error_y(), go_scatter3d_error_z(): Sets the stroke color of the error bars. Color: a CSS color string or R color name.

go_scatter3d_hoverlabel_font(), go_scatter3d_textfont(): Color: a CSS color string or R color name, or a vector of these.

go_scatter3d_legendgrouptitle_font(), go_scatter3d_line_colorbar_tickfont(), go_scatter3d_line_colorbar_title_font(), go_scatter3d_marker_colorbar_tickfont(), go_scatter3d_marker_colorbar_title_font(): Color: a CSS color string or R color name.

go_scatter3d_line(): Sets the line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘line.cmin‘ and ‘line.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scatter3d.line.colorscale, or a vector of these.

go_scatter3d_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.cmin‘ and ‘marker.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scatter3d.marker.colorscale, or a vector of these.

	`go_scatter3d_marker_line()`: Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by `scatter3d.marker.line.colorscale`, or a vector of these.
`copy_zstyle`	Logical.
`symmetric`	Determines whether or not the error bars have the same length in both direction (top/bottom for vertical bars, left/right for horizontal bars. Logical.
`thickness`	`go_scatter3d_error_x()`, `go_scatter3d_error_y()`, `go_scatter3d_error_z()`: Sets the thickness (in px) of the error bars. Number in the interval [0, Inf]. `go_scatter3d_line_colorbar()`, `go_scatter3d_marker_colorbar()`: Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].
`traceref`	Integer in the interval [0, Inf].
`tracerefminus`	Integer in the interval [0, Inf].
`type`	Determines the rule used to generate the error bars. If "constant", the bar lengths are of a constant value. Set this constant in ‘value‘. If "percent", the bar lengths correspond to a percentage of underlying data. Set this percentage in ‘value‘. If "sqrt", the bar lengths correspond to the square of the underlying data. If "data", the bar lengths are set with data set ‘array‘. One of `"percent"`, `"constant"`, `"sqrt"`, `"data"`.
`value`	`go_scatter3d_error_x()`, `go_scatter3d_error_y()`, `go_scatter3d_error_z()`: Sets the value of either the percentage (if ‘type‘ is set to "percent") or the constant (if ‘type‘ is set to "constant") corresponding to the lengths of the error bars. Number in the interval [0, Inf]. `go_scatter3d_line_colorbar_tickformatstop()`, `go_scatter3d_marker_colorbar_tickformats` string - dtickformat for described zoom level, the same as "tickformat" String.
`valueminus`	Sets the value of either the percentage (if ‘type‘ is set to "percent") or the constant (if ‘type‘ is set to "constant") corresponding to the lengths of the error bars in the bottom (left) direction for vertical (horizontal) bars Number in the interval [0, Inf].
`width`	`go_scatter3d_error_x()`, `go_scatter3d_error_y()`, `go_scatter3d_error_z()`: Sets the width (in px) of the cross-bar at both ends of the error bars. Number in the interval [0, Inf]. `go_scatter3d_line()`: Sets the line width (in px). Number in the interval [0, Inf]. `go_scatter3d_marker_line()`: Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf].
`align`	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.
`bgcolor`	`go_scatter3d_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these. `go_scatter3d_line_colorbar()`, `go_scatter3d_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

bordercolor
: go_scatter3d_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

  go_scatter3d_line_colorbar(), go_scatter3d_marker_colorbar(): Sets the axis line color. Color: a CSS color string or R color name.

font
: go_scatter3d_hoverlabel(): Sets the font used in hover labels. Named list or go_scatter3d_hoverlabel_font() object.

  go_scatter3d_legendgrouptitle(): Sets this legend group's title font. Named list or go_scatter3d_legendgrouptitle_font() object.

  go_scatter3d_line_colorbar_title(): Sets this color bar's title font. Named list or go_scatter3d_line_colorbar_title_font() object.

  go_scatter3d_marker_colorbar_title(): Sets this color bar's title font. Named list or go_scatter3d_marker_colorbar_title_font() object.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

family
: go_scatter3d_hoverlabel_font(), go_scatter3d_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

  go_scatter3d_legendgrouptitle_font(), go_scatter3d_line_colorbar_tickfont(), go_scatter3d_line_colorbar_title_font(), go_scatter3d_marker_colorbar_tickfont(), go_scatter3d_marker_colorbar_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: go_scatter3d_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

  go_scatter3d_legendgrouptitle_font(), go_scatter3d_line_colorbar_tickfont(), go_scatter3d_line_colorbar_title_font(), go_scatter3d_marker_colorbar_tickfont(), go_scatter3d_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow
: go_scatter3d_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_scatter3d_legendgrouptitle_font(), go_scatter3d_line_colorbar_tickfont(), go_scatter3d_line_colorbar_title_font(), go_scatter3d_marker_colorbar_tickfont(), go_scatter3d_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size

go_scatter3d_hoverlabel_font(), go_scatter3d_textfont(): Number in the interval [1, Inf], or a vector of these.

go_scatter3d_legendgrouptitle_font(), go_scatter3d_line_colorbar_tickfont(), go_scatter3d_line_colorbar_title_font(), go_scatter3d_marker_colorbar_tickfont(), go_scatter3d_marker_colorbar_title_font(): Number in the interval [1, Inf].

go_scatter3d_marker(): Sets the marker size (in px). Number in the interval [0, Inf], or a vector of these.

style

go_scatter3d_hoverlabel_font(), go_scatter3d_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_scatter3d_legendgrouptitle_font(), go_scatter3d_line_colorbar_tickfont(), go_scatter3d_line_colorbar_title_font(), go_scatter3d_marker_colorbar_tickfont(), go_scatter3d_marker_colorbar_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase

go_scatter3d_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_scatter3d_legendgrouptitle_font(), go_scatter3d_line_colorbar_tickfont(), go_scatter3d_line_colorbar_title_font(), go_scatter3d_marker_colorbar_tickfont(), go_scatter3d_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant

go_scatter3d_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_scatter3d_legendgrouptitle_font(), go_scatter3d_line_colorbar_tickfont(), go_scatter3d_line_colorbar_title_font(), go_scatter3d_marker_colorbar_tickfont(), go_scatter3d_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

go_scatter3d_textfont(): Sets the variant of the font. One of "normal", "small-caps", or a vector of these.

weight

go_scatter3d_hoverlabel_font(), go_scatter3d_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_scatter3d_legendgrouptitle_font(), go_scatter3d_line_colorbar_tickfont(), go_scatter3d_line_colorbar_title_font(), go_scatter3d_marker_colorbar_tickfont(), go_scatter3d_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

`autocolorscale` `go_scatter3d_line()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘line.colorscale‘. Has an effect only if in ‘line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`go_scatter3d_marker()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`go_scatter3d_marker_line()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`cauto` `go_scatter3d_line()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘line.color‘) or the bounds set in ‘line.cmin‘ and ‘line.cmax‘ Has an effect only if in ‘line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘line.cmin‘ and ‘line.cmax‘ are set by the user. Logical.

`go_scatter3d_marker()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

`go_scatter3d_marker_line()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

`cmax` `go_scatter3d_line()`: Sets the upper bound of the color domain. Has an effect only if in ‘line.color‘ is set to a numerical array. Value should have the same units as in ‘line.color‘ and if set, ‘line.cmin‘ must be set as well. Number.

`go_scatter3d_marker()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

`go_scatter3d_marker_line()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

`cmid` `go_scatter3d_line()`: Sets the mid-point of the color domain by scaling ‘line.cmin‘ and/or ‘line.cmax‘ to be equidistant to this point. Has an effect only if in

'line.color' is set to a numerical array. Value should have the same units as in 'line.color'. Has no effect when 'line.cauto' is 'false'. Number.

`go_scatter3d_marker()`: Sets the mid-point of the color domain by scaling 'marker.cmin' and/or 'marker.cmax' to be equidistant to this point. Has an effect only if in 'marker.color' is set to a numerical array. Value should have the same units as in 'marker.color'. Has no effect when 'marker.cauto' is 'false'. Number.

`go_scatter3d_marker_line()`: Sets the mid-point of the color domain by scaling 'marker.line.cmin' and/or 'marker.line.cmax' to be equidistant to this point. Has an effect only if in 'marker.line.color' is set to a numerical array. Value should have the same units as in 'marker.line.color'. Has no effect when 'marker.line.cauto' is 'false'. Number.

`cmin`

`go_scatter3d_line()`: Sets the lower bound of the color domain. Has an effect only if in 'line.color' is set to a numerical array. Value should have the same units as in 'line.color' and if set, 'line.cmax' must be set as well. Number.

`go_scatter3d_marker()`: Sets the lower bound of the color domain. Has an effect only if in 'marker.color' is set to a numerical array. Value should have the same units as in 'marker.color' and if set, 'marker.cmax' must be set as well. Number.

`go_scatter3d_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in 'marker.line.color' is set to a numerical array. Value should have the same units as in 'marker.line.color' and if set, 'marker.line.cmax' must be set as well. Number.

`coloraxis`

Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under 'layout.coloraxis', 'layout.coloraxis2', etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar`

`go_scatter3d_line()`: Named list or `go_scatter3d_line_colorbar()` object.

`go_scatter3d_marker()`: Named list or `go_scatter3d_marker_colorbar()` object.

`colorscale`

`go_scatter3d_line()`: Sets the colorscale. Has an effect only if in 'line.color' is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the bounds of the colorscale in color space, use 'line.cmin' and 'line.cmax'. Alternatively, 'colorscale' may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_scatter3d_marker()`: Sets the colorscale. Has an effect only if in 'marker.color' is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the

bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Re
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_scatter3d_marker_line()`: Sets the colorscale. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.line.cmin‘ and ‘marker.line.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,H
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`dash` Sets the dash style of the lines. One of `"dash"`, `"dashdot"`, `"dot"`, `"longdash"`, `"longdashdot"`, `"solid"`.

`reversescale` `go_scatter3d_line()`: Reverses the color mapping if true. Has an effect only if in ‘line.color‘ is set to a numerical array. If true, ‘line.cmin‘ will correspond to the last color in the array and ‘line.cmax‘ will correspond to the first color. Logical.

`go_scatter3d_marker()`: Reverses the color mapping if true. Has an effect only if in ‘marker.color‘ is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.

`go_scatter3d_marker_line()`: Reverses the color mapping if true. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. If true, ‘marker.line.cmin‘ will correspond to the last color in the array and ‘marker.line.cmax‘ will correspond to the first color. Logical.

`showscale` `go_scatter3d_line()`: Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘line.color‘ is set to a numerical array. Logical.

`go_scatter3d_marker()`: Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘marker.color‘ is set to a numerical array. Logical.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended"

covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands`
If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date

data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` `go_scatter3d_line_colorbar()`: Sets the color bar's tick label font Named list or `go_scatter3d_line_colorbar_tickfont()` object.

`go_scatter3d_marker_colorbar()`: Sets the color bar's tick label font Named list or `go_scatter3d_marker_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops`

`go_scatter3d_line_colorbar()`: List of named lists or `go_scatter3d_line_colorbar_tickformats` objects.

`go_scatter3d_marker_colorbar()`: List of named lists or `go_scatter3d_marker_colorbar_tickfor` objects.

`ticklabeloverflow`

Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition`

Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

tickprefix Sets a tick label prefix. String.

ticks Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of "outside", "inside", "".

ticksuffix Sets a tick label suffix. String.

ticktext Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth Sets the tick width (in px). Number in the interval [0, Inf].

title go_scatter3d_line_colorbar(): String (the title text), named list or go_scatter3d_line_colorbar_ object.

go_scatter3d_marker_colorbar(): String (the title text), named list or go_scatter3d_marker_colorb object.

xanchor Sets this color bar's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the color bar. Defaults to "left" when 'orientation' is "v" and "center" when 'orientation' is "h". One of "left", "center", "right".

xpad Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref Sets the container 'x' refers to. "container" spans the entire 'width' of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".

yanchor Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of "top", "middle", "bottom".

ypad Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

dtickrange range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

templateitemname

Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled:

false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`side` Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.

`sizemin` Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the minimum size (in px) of the rendered marker points. Number in the interval [0, Inf].

`sizemode` Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the rule for which the data in ‘size‘ is converted to pixels. One of `"diameter"`, `"area"`.

`sizeref` Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the scale factor used to determine the rendered size of marker points. Use with ‘sizemin‘ and ‘sizemode‘. Number.

`symbol` Sets the marker symbol type. One of `"circle"`, `"circle-open"`, `"cross"`, `"diamond"`, `"diamond-open"`, `"square"`, `"square-open"`, `"x"`, or a vector of these.

`scale` Sets the scale factor determining the size of the projection marker points. Number in the interval [0, 10].

`show` `go_scatter3d_projection_x()`: Sets whether or not projections are shown along the x axis. Logical.

`go_scatter3d_projection_y()`: Sets whether or not projections are shown along the y axis. Logical.

`go_scatter3d_projection_z()`: Sets whether or not projections are shown along the z axis. Logical.

Details

`go_scatter3d()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_scatter3d()` creates an object and adds it to a figure.

Nested objects documented here:

`go_scatter3d_error_x()` `scatter3d.error_x`

`go_scatter3d_error_y()` `scatter3d.error_y`

`go_scatter3d_error_z()` `scatter3d.error_z`

`go_scatter3d_hoverlabel()` `scatter3d.hoverlabel`

`go_scatter3d_hoverlabel_font()` `scatter3d.hoverlabel.font`: Sets the font used in hover labels.

`go_scatter3d_legendgrouptitle()` `scatter3d.legendgrouptitle`

`go_scatter3d_legendgrouptitle_font()` `scatter3d.legendgrouptitle.font`: Sets this legend group's title font.

`go_scatter3d_line()` `scatter3d.line`

`go_scatter3d_line_colorbar()` `scatter3d.line.colorbar`

`go_scatter3d_line_colorbar_tickfont()` `scatter3d.line.colorbar.tickfont`: Sets the color bar's tick label font

`go_scatter3d_line_colorbar_tickformatstop()` `scatter3d.line.colorbar.tickformatstop`

`go_scatter3d_line_colorbar_title()` `scatter3d.line.colorbar.title`

`go_scatter3d_line_colorbar_title_font()` `scatter3d.line.colorbar.title.font`: Sets this color bar's title font.

`go_scatter3d_marker()` `scatter3d.marker`

`go_scatter3d_marker_colorbar()` `scatter3d.marker.colorbar`

`go_scatter3d_marker_colorbar_tickfont()` `scatter3d.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_scatter3d_marker_colorbar_tickformatstop()` `scatter3d.marker.colorbar.tickformatstop`

`go_scatter3d_marker_colorbar_title()` `scatter3d.marker.colorbar.title`

`go_scatter3d_marker_colorbar_title_font()` `scatter3d.marker.colorbar.title.font`: Sets this color bar's title font.

`go_scatter3d_marker_line()` `scatter3d.marker.line`

`go_scatter3d_projection()` `scatter3d.projection`

`go_scatter3d_projection_x()` `scatter3d.projection.x`

`go_scatter3d_projection_y()` `scatter3d.projection.y`

`go_scatter3d_projection_z()` `scatter3d.projection.z`

`go_scatter3d_textfont()` `scatter3d.textfont`: Sets the text font.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_scatter3d(x = 1:3, y = c(2, 1, 3), mode = "markers", marker_color = "red")
```

---

add_scattercarpet *Scattercarpet traces*

Description

Plots a scatter trace on either the first carpet axis or the carpet axis with a matching 'carpet' attribute.

Usage

```
add_scattercarpet(fig, ...)

## Default S3 method:
add_scattercarpet(
  fig,
  ...,
  data = NULL,
  a = NULL,
  b = NULL,
  carpet = NULL,
  connectgaps = NULL,
  customdata = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  xaxis = NULL,
  yaxis = NULL,
  zorder = NULL,
```

```
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_scattercarpet(
  arg = NULL,
  ...,
  a = NULL,
  b = NULL,
  carpet = NULL,
  connectgaps = NULL,
  customdata = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  xaxis = NULL,
  yaxis = NULL,
```

```
  zorder = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
```

```
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_line(
  arg = NULL,
  ...,
  backoff = NULL,
  color = NULL,
  dash = NULL,
  shape = NULL,
  smoothing = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_marker(
  arg = NULL,
  ...,
  angle = NULL,
  angleref = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  gradient = NULL,
  line = NULL,
  maxdisplayed = NULL,
  opacity = NULL,
  reversescale = NULL,
  showscale = NULL,
  size = NULL,
  sizemin = NULL,
  sizemode = NULL,
  sizeref = NULL,
  standoff = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_marker_colorbar(
```

```
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
```

```
  skip_invalid = FALSE
)

go_scattercarpet_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
```

```
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_marker_gradient(
  arg = NULL,
  ...,
  color = NULL,
  type = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_marker_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  dash = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_selected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_selected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_selected_textfont(
```

```
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scattercarpet_unselected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.

`...`	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
`data`	A data frame in which formula properties (e.g. `x = ~col`, `marker_size = ~col`) are evaluated. Default: the data given to `go_figure()`. Only the evaluated values are added to the trace. A `crosstalk::SharedData` works the same way, and links the trace to its crosstalk group (see `highlight()`).
`a`	Sets the a-axis coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
`b`	Sets the b-axis coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
`carpet`	An identifier for this carpet, so that 'scattercarpet' and 'contourcarpet' traces can specify a carpet plot on which they lie String.
`connectgaps`	Determines whether or not gaps (i.e. {nan} or missing values) in the provided data arrays are connected. Logical.
`customdata`	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
`fill`	Sets the area to fill with a solid color. Use with 'fillcolor' if not "none". scatterternary has a subset of the options available to scatter. "toself" connects the endpoints of the trace (or each segment of the trace if it has gaps) into a closed shape. "tonext" fills the space between two traces if one completely encloses the other (eg consecutive contour lines), and behaves like "toself" if there is no trace before it. "tonext" should not be used if one trace does not enclose the other. One of `"none"`, `"toself"`, `"tonext"`.
`fillcolor`	Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name.
`hoverinfo`	Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of `"a"`, `"b"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.
`hoverlabel`	Named list or `go_scattercarpet_hoverlabel()` object.
`hoveron`	Do the hover effects highlight individual points (markers or line points) or do they highlight filled regions? If the fill is "toself" or "tonext" and there are no markers or text, then the default is "fills", otherwise it is "points". Any combination of `"points"`, `"fills"` joined with `"+"`.
`hovertemplate`	Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for

example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each (a,b) point. If a single string, the same string appears over all the data points. If an array of strings, the items are mapped in order to the data points in (a,b). To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_scattercarpet_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line`
: `go_scattercarpet()`: Named list or `go_scattercarpet_line()` object.

`go_scattercarpet_marker()`: Named list or `go_scattercarpet_marker_line()` object.

`marker` `go_scattercarpet()`: Named list or `go_scattercarpet_marker()` object.

`go_scattercarpet_selected()`: Named list or `go_scattercarpet_selected_marker()` object.

`go_scattercarpet_unselected()`: Named list or `go_scattercarpet_unselected_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`mode` Determines the drawing mode for this scatter trace. If the provided ‘mode‘ includes "text" then the ‘text‘ elements appear at the coordinates. Otherwise, the ‘text‘ elements appear on hover. If there are less than 20 points and the trace is not stacked then the default is "lines+markers". Otherwise, "lines". Any combination of `"lines"`, `"markers"`, `"text"` joined with `"+"`, or one of `"none"`.

`name` `go_scattercarpet()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_scattercarpet_marker_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` `go_scattercarpet()`: Sets the opacity of the trace. Number in the interval [0, 1].

`go_scattercarpet_marker()`: Sets the marker opacity. Number in the interval [0, 1], or a vector of these.

`go_scattercarpet_selected_marker()`: Sets the marker opacity of selected points. Number in the interval [0, 1].

`go_scattercarpet_unselected_marker()`: Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

`selected` Named list or `go_scattercarpet_selected()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`text` `go_scattercarpet()`: Sets text elements associated with each (a,b) point. If a single string, the same string appears over all the data points. If an array

of strings, the items are mapped in order to the data points in (a,b). If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_scattercarpet_legendgrouptitle()`: Sets the title of the legend group. String.

`go_scattercarpet_marker_colorbar_title()`: Sets the title of the color bar. String.

`textfont` `go_scattercarpet()`: Sets the text font. Named list or `go_scattercarpet_textfont()` object.

`go_scattercarpet_selected()`: Named list or `go_scattercarpet_selected_textfont()` object.

`go_scattercarpet_unselected()`: Named list or `go_scattercarpet_unselected_textfont()` object.

`textposition` Sets the positions of the ‘text‘ elements with respects to the (x,y) coordinates. One of `"top left"`, `"top center"`, `"top right"`, `"middle left"`, `"middle center"`, `"middle right"`, `"bottom left"`, `"bottom center"`, `"bottom right"`, or a vector of these.

`texttemplate` Template string used for rendering the information text that appears on points. Note that this will override ‘textinfo‘. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format’s syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format’s syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can’t be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variables ‘a‘, ‘b‘ and ‘text‘. String, or a vector of these.

`texttemplatefallback`

Fallback string that’s displayed when a variable referenced in a template is missing. If the boolean value ’false’ is passed in, the specifier with the missing variable will be displayed. Any value.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(x|y)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace

has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

`unselected` Named list or `go_scattercarpet_unselected()` object.

`visible` Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`xaxis` Sets a reference between this trace's x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to 'layout.xaxis'. If "x2", the x coordinates refer to 'layout.xaxis2', and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

`yaxis` Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to 'layout.yaxis'. If "y2", the y coordinates refer to 'layout.yaxis2', and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...

`zorder` Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher 'zorder' appear in front of those with lower 'zorder'. Integer.

`row` Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`col` Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`secondary_y` If `TRUE`, use the secondary y axis of the subplot.

`skip_invalid` If `TRUE`, silently drop invalid properties instead of raising an error.

`arg` Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

`align` Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`bgcolor` `go_scattercarpet_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`go_scattercarpet_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

`bordercolor` `go_scattercarpet_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`go_scattercarpet_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

`font` `go_scattercarpet_hoverlabel()`: Sets the font used in hover labels. Named list or `go_scattercarpet_hoverlabel_font()` object.

`go_scattercarpet_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_scattercarpet_legendgrouptitle_font()` object.

`go_scattercarpet_marker_colorbar_title()`: Sets this color bar's title font. Named list or `go_scattercarpet_marker_colorbar_title_font()` object.

`namelength` Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows

the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color `go_scattercarpet_hoverlabel_font()`, `go_scattercarpet_textfont()`: Color: a CSS color string or R color name, or a vector of these.

`go_scattercarpet_legendgrouptitle_font()`, `go_scattercarpet_marker_colorbar_tickfont()` `go_scattercarpet_marker_colorbar_title_font()`: Color: a CSS color string or R color name.

`go_scattercarpet_line()`: Sets the line color. Color: a CSS color string or R color name.

`go_scattercarpet_marker()`: Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.cmin' and 'marker.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by `scattercarpet.marker.colorscale`, or a vector of these.

`go_scattercarpet_marker_gradient()`: Sets the final color of the gradient fill: the center color for radial, the right for horizontal, or the bottom for vertical. Color: a CSS color string or R color name, or a vector of these.

`go_scattercarpet_marker_line()`: Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.line.cmin' and 'marker.line.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by `scattercarpet.marker.line.colorscale`, or a vector of these.

`go_scattercarpet_selected_marker()`: Sets the marker color of selected points. Color: a CSS color string or R color name.

`go_scattercarpet_selected_textfont()`: Sets the text font color of selected points. Color: a CSS color string or R color name.

`go_scattercarpet_unselected_marker()`: Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

`go_scattercarpet_unselected_textfont()`: Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

family `go_scattercarpet_hoverlabel_font()`, `go_scattercarpet_textfont()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

`go_scattercarpet_legendgrouptitle_font()`, `go_scattercarpet_marker_colorbar_tickfont()` `go_scattercarpet_marker_colorbar_title_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font

families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition

go_scattercarpet_hoverlabel_font(), go_scattercarpet_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_scattercarpet_legendgrouptitle_font(), go_scattercarpet_marker_colorbar_tickfont() go_scattercarpet_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow

go_scattercarpet_hoverlabel_font(), go_scattercarpet_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_scattercarpet_legendgrouptitle_font(), go_scattercarpet_marker_colorbar_tickfont() go_scattercarpet_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size

go_scattercarpet_hoverlabel_font(), go_scattercarpet_textfont(): Number in the interval [1, Inf], or a vector of these.

go_scattercarpet_legendgrouptitle_font(), go_scattercarpet_marker_colorbar_tickfont() go_scattercarpet_marker_colorbar_title_font(): Number in the interval [1, Inf].

go_scattercarpet_marker(): Sets the marker size (in px). Number in the interval [0, Inf], or a vector of these.

go_scattercarpet_selected_marker(): Sets the marker size of selected points. Number in the interval [0, Inf].

go_scattercarpet_unselected_marker(): Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

style

go_scattercarpet_hoverlabel_font(), go_scattercarpet_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_scattercarpet_legendgrouptitle_font(), go_scattercarpet_marker_colorbar_tickfont() go_scattercarpet_marker_colorbar_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase

go_scattercarpet_hoverlabel_font(), go_scattercarpet_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_scattercarpet_legendgrouptitle_font(), go_scattercarpet_marker_colorbar_tickfont() go_scattercarpet_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_scattercarpet_hoverlabel_font(),go_scattercarpet_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.
go_scattercarpet_legendgrouptitle_font(),go_scattercarpet_marker_colorbar_tickfont() go_scattercarpet_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_scattercarpet_hoverlabel_font(),go_scattercarpet_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.
go_scattercarpet_legendgrouptitle_font(),go_scattercarpet_marker_colorbar_tickfont() go_scattercarpet_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

backoff
: Sets the line back off from the end point of the nth line segment (in px). This option is useful e.g. to avoid overlap with arrowhead markers. With "auto" the lines would trim before markers if ‘marker.angleref‘ is set to "previous". Number in the interval [0, Inf], or a vector of these.

dash
: go_scattercarpet_line(): Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\
go_scattercarpet_marker_line(): Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$", or a vector of these.

shape
: Determines the line shape. With "spline" the lines are drawn using spline interpolation. The other available values correspond to step-wise line shapes. One of "linear", "spline".

smoothing
: Has an effect only if ‘shape‘ is set to "spline" Sets the amount of smoothing. 0 corresponds to no smoothing (equivalent to a "linear" shape). Number in the interval [0, 1.3].

width
: go_scattercarpet_line(): Sets the line width (in px). Number in the interval [0, Inf].
go_scattercarpet_marker_line(): Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

angle
: Sets the marker angle in respect to ‘angleref‘. Angle in degrees, or a vector of these.

angleref
: Sets the reference for marker angle. With "previous", angle 0 points along the line from the previous point to this one. With "up", angle 0 points toward the top of the screen. One of "previous", "up".

autocolorscale
: go_scattercarpet_marker(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be

chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`go_scattercarpet_marker_line()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`cauto` `go_scattercarpet_marker()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

`go_scattercarpet_marker_line()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

`cmax` `go_scattercarpet_marker()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

`go_scattercarpet_marker_line()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

`cmid` `go_scattercarpet_marker()`: Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

`go_scattercarpet_marker_line()`: Sets the mid-point of the color domain by scaling ‘marker.line.cmin‘ and/or ‘marker.line.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

`cmin` `go_scattercarpet_marker()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`go_scattercarpet_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes

are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: "coloraxis", "coloraxis2", "coloraxis3", ...

colorbar — Named list or go_scattercarpet_marker_colorbar() object.

colorscale — go_scattercarpet_marker(): Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rai
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

go_scattercarpet_marker_line(): Sets the colorscale. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.line.cmin‘ and ‘marker.line.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,H
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

gradient — Named list or go_scattercarpet_marker_gradient() object.

maxdisplayed — Sets a maximum number of points to be drawn on the graph. 0 corresponds to no limit. Number in the interval [0, Inf].

reversescale — go_scattercarpet_marker(): Reverses the color mapping if true. Has an effect only if in ‘marker.color‘ is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.

go_scattercarpet_marker_line(): Reverses the color mapping if true. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. If true, ‘marker.line.cmin‘ will correspond to the last color in the array and ‘marker.line.cmax‘ will correspond to the first color. Logical.

showscale — Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘marker.color‘ is set to a numerical array. Logical.

sizemin — Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the minimum size (in px) of the rendered marker points. Number in the interval [0, Inf].

sizemode — Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the rule for which the data in ‘size‘ is converted to pixels. One of "diameter", "area".

sizeref — Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the scale factor used to determine the rendered size of marker points. Use with ‘sizemin‘ and ‘sizemode‘. Number.

standoff Moves the marker away from the data point in the direction of 'angle' (in px). This can be useful for example if you have another marker at this location and you want to point an arrowhead marker at it. Number in the interval [0, Inf], or a vector of these.

symbol Sets the marker symbol type. Adding 100 is equivalent to appending "-open" to a symbol name. Adding 200 is equivalent to appending "-dot" to a symbol name. Adding 300 is equivalent to appending "-open-dot" or "dot-open" to a symbol name. One of "0", "circle", "100", "circle-open", "200", "circle-dot", "300", "circle-open-dot", "1", "square", "101", "square-open", "201", "square-dot", "301", "square-open-dot", "2", "diamond", "102", "diamond-open", "202", "diamond-dot", "302", "diamond-open-dot", "3", "cross", "103", "cross-open", "203", "cross-dot", "303", "cross-open-dot", "4", "x", "104", "x-open", "204", "x-dot", "304", "x-open-dot", "5", "triangle-up", "105", "triangle-up-open", "205", "triangle-up-dot", "305", "triangle-up-open-dot", "6", "triangle-down", "106", "triangle-down-open", "206", "triangle-down-dot", "306", "triangle-down-open-dot", "7", "triangle-left", "107", "triangle-left-open", "207", "triangle-left-dot", "307", "triangle-left-open-dot", "8", "triangle-right", "108", "triangle-right-open", "208", "triangle-right-dot", "308", "triangle-right-open-do "9", "triangle-ne", "109", "triangle-ne-open", "209", "triangle-ne-dot", "309", "triangle-ne-open-dot", "10", "triangle-se", "110", "triangle-se-open", "210", "triangle-se-dot", "310", "triangle-se-open-dot", "11", "triangle-sw", "111", "triangle-sw-open", "211", "triangle-sw-dot", "311", "triangle-sw-open-dot", "12", "triangle-nw", "112", "triangle-nw-open", "212", "triangle-nw-dot", "312", "triangle-nw-open-dot", "13", "pentagon", "113", "pentagon-open", "213", "pentagon-dot", "313", "pentagon-open-dot", "14", "hexagon", "114", "hexagon-open", "214", "hexagon-dot", "314", "hexagon-open-dot", "15", "hexagon2", "115", "hexagon2-open", "215", "hexagon2-dot", "315", "hexagon2-open-dot", "16", "octagon", "116", "octagon-open", "216", "octagon-dot", "316", "octagon-open-dot", "17", "star", "117", "star-open", "217", "star-dot", "317", "star-open-dot", "18", "hexagram", "118", "hexagram-open", "218", "hexagram-dot", "318", "hexagram-open-dot", "19", "star-triangle-up", "119", "star-triangle-up-open", "219", "star-triangle-up-dot", "319", "star-triangle-up-open-dot", "20", "star-triangle-down", "120", "star-triangle-down-open "220", "star-triangle-down-dot", "320", "star-triangle-down-open-dot", "21", "star-square", "121", "star-square-open", "221", "star-square-dot", "321", "star-square-open-dot", "22", "star-diamond", "122", "star-diamond-open", "222", "star-diamond-dot", "322", "star-diamond-open-dot", "23", "diamond-tall", "123", "diamond-tall-open", "223", "diamond-tall-dot", "323", "diamond-tall-open-dot", "24", "diamond-wide", "124", "diamond-wide-open", "224", "diamond-wide-dot", "324", "diamond-wide-open-dot", "25", "hourglass", "125", "hourglass-open", "26", "bowtie", "126", "bowtie-open", "27", "circle-cross", "127", "circle-cross-open", "28", "circle-x", "128", "circle-x-open", "29", "square-cross", "129", "square-cross-open", "30", "square-x", "130", "square-x-open", "31", "diamond-cross", "131", "diamond-cross-open", "32", "diamond-x", "132", "diamond-x-open", "33", "cross-thin", "133", "cross-thin-open", "34", "x-thin", "134", "x-thin-open", "35", "asterisk", "135", "asterisk-open", "36", "hash", "136", "hash-open", "236", "hash-dot", "336", "hash-open-dot", "37", "y-up", "137", "y-up-open", "38", "y-down", "138", "y-down-open",

"39", "y-left", "139", "y-left-open", "40", "y-right", "140", "y-right-open", "41", "line-ew", "141", "line-ew-open", "42", "line-ns", "142", "line-ns-open", "43", "line-ne", "143", "line-ne-open", "44", "line-nw", "144", "line-nw-open", "45", "arrow-up", "145", "arrow-up-open", "46", "arrow-down", "146", "arrow-down-open", "47", "arrow-left", "147", "arrow-left-open", "48", "arrow-right", "148", "arrow-right-open", "49", "arrow-bar-up", "149", "arrow-bar-up-open", "50", "arrow-bar-down", "150", "arrow-bar-down-open", "51", "arrow-bar-left", "151", "arrow-bar-left-open", "52", "arrow-bar-right", "152", "arrow-bar-right-open", "53", "arrow", "153", "arrow-open", "54", "arrow-wide", "154", "arrow-wide-open", 0, 100, 200, 300, 1, 101, 201, 301, 2, 102, 202, 302, 3, 103, 203, 303, 4, 104, 204, 304, 5, 105, 205, 305, 6, 106, 206, 306, 7, 107, 207, 307, 8, 108, 208, 308, 9, 109, 209, 309, 10, 110, 210, 310, 11, 111, 211, 311, 12, 112, 212, 312, 13, 113, 213, 313, 14, 114, 214, 314, 15, 115, 215, 315, 16, 116, 216, 316, 17, 117, 217, 317, 18, 118, 218, 318, 19, 119, 219, 319, 20, 120, 220, 320, 21, 121, 221, 321, 22, 122, 222, 322, 23, 123, 223, 323, 24, 124, 224, 324, 25, 125, 26, 126, 27, 127, 28, 128, 29, 129, 30, 130, 31, 131, 32, 132, 33, 133, 34, 134, 35, 135, 36, 136, 236, 336, 37, 137, 38, 138, 39, 139, 40, 140, 41, 141, 42, 142, 43, 143, 44, 144, 45, 145, 46, 146, 47, 147, 48, 148, 49, 149, 50, 150, 51, 151, 52, 152, 53, 153, 54, 154, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of "fraction", "pixels".

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar’s tick label font Named list or `go_scattercarpet_marker_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops
: List of named lists or go_scattercarpet_marker_colorbar_tickformatstop() objects.

ticklabeloverflow
: Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelstep
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

ticklen
: Sets the tick length (in px). Number in the interval [0, Inf].

tickmode
: Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of "auto", "linear", "array".

tickprefix
: Sets a tick label prefix. String.

ticks
: Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of "outside", "inside", "".

ticksuffix
: Sets a tick label suffix. String.

ticktext
: Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals
: Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth
: Sets the tick width (in px). Number in the interval [0, Inf].

title
: String (the title text), named list or go_scattercarpet_marker_colorbar_title() object.

x
: Sets the x position with respect to 'xref' of the color bar (in plot fraction). When 'xref' is "paper", defaults to 1.02 when 'orientation' is "v" and 0.5 when 'orientation' is "h". When 'xref' is "container", defaults to 1 when 'orientation' is "v" and 0.5 when 'orientation' is "h". Must be between 0 and 1 if 'xref' is "container" and between "-2" and 3 if 'xref' is "paper". Number.

`xanchor` Sets this color bar's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the color bar. Defaults to "left" when 'orientation' is "v" and "center" when 'orientation' is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container 'x' refers to. "container" spans the entire 'width' of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`y` Sets the y position with respect to 'yref' of the color bar (in plot fraction). When 'yref' is "paper", defaults to 0.5 when 'orientation' is "v" and 1.02 when 'orientation' is "h". When 'yref' is "container", defaults to 0.5 when 'orientation' is "v" and 1 when 'orientation' is "h". Must be between 0 and 1 if 'yref' is "container" and between "-2" and 3 if 'yref' is "paper". Number.

`yanchor` Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of `"top"`, `"middle"`, `"bottom"`.

`ypad` Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

`yref` Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

`dtickrange` range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

`enabled` Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

`templateitemname` Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

`value` string - dtickformat for described zoom level, the same as "tickformat" String.

`side` Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of `"right"`, `"top"`, `"bottom"`.

`type` Sets the type of gradient used to fill the markers One of `"radial"`, `"horizontal"`, `"vertical"`, `"none"`, or a vector of these.

Details

`go_scattercarpet()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_scattercarpet()` creates an object and adds it to a figure.

Nested objects documented here:

`go_scattercarpet_hoverlabel()` `scattercarpet.hoverlabel`

`go_scattercarpet_hoverlabel_font()` `scattercarpet.hoverlabel.font`: Sets the font used in hover labels.

`go_scattercarpet_legendgrouptitle()` `scattercarpet.legendgrouptitle`

`go_scattercarpet_legendgrouptitle_font()` `scattercarpet.legendgrouptitle.font`: Sets this legend group's title font.

`go_scattercarpet_line()` `scattercarpet.line`

`go_scattercarpet_marker()` `scattercarpet.marker`

`go_scattercarpet_marker_colorbar()` `scattercarpet.marker.colorbar`

`go_scattercarpet_marker_colorbar_tickfont()` `scattercarpet.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_scattercarpet_marker_colorbar_tickformatstop()` `scattercarpet.marker.colorbar.tickformatstop`

`go_scattercarpet_marker_colorbar_title()` `scattercarpet.marker.colorbar.title`

`go_scattercarpet_marker_colorbar_title_font()` `scattercarpet.marker.colorbar.title.font`: Sets this color bar's title font.

`go_scattercarpet_marker_gradient()` `scattercarpet.marker.gradient`

`go_scattercarpet_marker_line()` `scattercarpet.marker.line`

`go_scattercarpet_selected()` `scattercarpet.selected`

`go_scattercarpet_selected_marker()` `scattercarpet.selected.marker`

`go_scattercarpet_selected_textfont()` `scattercarpet.selected.textfont`

`go_scattercarpet_textfont()` `scattercarpet.textfont`: Sets the text font.

`go_scattercarpet_unselected()` `scattercarpet.unselected`

`go_scattercarpet_unselected_marker()` `scattercarpet.unselected.marker`

`go_scattercarpet_unselected_textfont()` `scattercarpet.unselected.textfont`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_scattercarpet(mode = "markers", marker_color = "red")
```

add_scattergeo *Scattergeo traces*

Description

The data visualized as scatter point or lines on a geographic map is provided either by longitude/latitude pairs in 'lon' and 'lat' respectively or by geographic location IDs or names in 'locations'.

Usage

```
add_scattergeo(fig, ...)

## Default S3 method:
add_scattergeo(
  fig,
  ...,
  data = NULL,
  connectgaps = NULL,
  customdata = NULL,
  featureidkey = NULL,
  fill = NULL,
  fillcolor = NULL,
  geo = NULL,
  geojson = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  lat = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  locationmode = NULL,
  locations = NULL,
  lon = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
```

```
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_scattergeo(
  arg = NULL,
  ...,
  connectgaps = NULL,
  customdata = NULL,
  featureidkey = NULL,
  fill = NULL,
  fillcolor = NULL,
  geo = NULL,
  geojson = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  lat = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  locationmode = NULL,
  locations = NULL,
  lon = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
```

```
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_scattergeo_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_scattergeo_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattergeo_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)
```

```
  go_scattergeo_legendgrouptitle_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_scattergeo_line(
    arg = NULL,
    ...,
    color = NULL,
    dash = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_scattergeo_marker(
    arg = NULL,
    ...,
    angle = NULL,
    angleref = NULL,
    autocolorscale = NULL,
    cauto = NULL,
    cmax = NULL,
    cmid = NULL,
    cmin = NULL,
    color = NULL,
    coloraxis = NULL,
    colorbar = NULL,
    colorscale = NULL,
    gradient = NULL,
    line = NULL,
    opacity = NULL,
    reversescale = NULL,
    showscale = NULL,
    size = NULL,
    sizemin = NULL,
    sizemode = NULL,
    sizeref = NULL,
```

```
  standoff = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_scattergeo_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
```

```
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_scattergeo_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattergeo_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_scattergeo_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scattergeo_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
```

```
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattergeo_marker_gradient(
  arg = NULL,
  ...,
  color = NULL,
  type = NULL,
  skip_invalid = FALSE
)

go_scattergeo_marker_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  dash = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scattergeo_selected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scattergeo_selected_marker(
  arg = NULL,
  ...,
  color = NULL,
```

```
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scattergeo_selected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

go_scattergeo_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattergeo_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scattergeo_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scattergeo_unselected_textfont(
  arg = NULL,
  ...,
  color = NULL,
```

```
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
connectgaps	Determines whether or not gaps (i.e. {nan} or missing values) in the provided data arrays are connected. Logical.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
featureidkey	Sets the key in GeoJSON features which is used as id to match the items included in the 'locations' array. Only has an effect when 'geojson' is set. Support nested property, for example "properties.name". String.
fill	Sets the area to fill with a solid color. Use with 'fillcolor' if not "none". "toself" connects the endpoints of the trace (or each segment of the trace if it has gaps) into a closed shape. One of "none", "toself".
fillcolor	Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name.
geo	Sets a reference between this trace's geospatial coordinates and a geographic map. If "geo" (the default value), the geospatial coordinates refer to 'layout.geo'. If "geo2", the geospatial coordinates refer to 'layout.geo2', and so on. Subplot id: "geo", "geo2", "geo3", ...
geojson	Sets optional GeoJSON data associated with this trace. If not given, the features on the base map are used when 'locations' is set. It can be set as a valid GeoJSON object or as a URL string. Note that we only accept GeoJSONs of type "FeatureCollection" or "Feature" with geometries of type "Polygon" or "MultiPolygon". Any value.
hoverinfo	Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of "lon", "lat", "location", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_scattergeo_hoverlabel() object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_},

{%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each (lon,lat) pair or item in 'locations'. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (lon,lat) or 'locations' coordinates. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`lat`
: Sets the latitude coordinates (in degrees North). Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_scattergeo_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line` `go_scattergeo()`: Named list or `go_scattergeo_line()` object.

`go_scattergeo_marker()`: Named list or `go_scattergeo_marker_line()` object.

`locationmode` Determines the set of locations used to match entries in 'locations' to regions on the map. Values "ISO-3", "USA-states", "country names" correspond to features on the base map and value "geojson-id" corresponds to features from a custom GeoJSON linked to the 'geojson' attribute. "USA-states" accepts both two-letter abbreviations (e.g. "CA") and full state names (e.g. "California"). One of `"ISO-3"`, `"USA-states"`, `"country names"`, `"geojson-id"`.

`locations` Sets the coordinates via location IDs or names. Coordinates correspond to the centroid of each location given. See 'locationmode' for more info. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`lon` Sets the longitude coordinates (in degrees East). Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`marker` `go_scattergeo()`: Named list or `go_scattergeo_marker()` object.

`go_scattergeo_selected()`: Named list or `go_scattergeo_selected_marker()` object.

`go_scattergeo_unselected()`: Named list or `go_scattergeo_unselected_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

`mode` Determines the drawing mode for this scatter trace. If the provided 'mode' includes "text" then the 'text' elements appear at the coordinates. Otherwise, the 'text' elements appear on hover. If there are less than 20 points and the trace is not stacked then the default is "lines+markers". Otherwise, "lines". Any combination of `"lines"`, `"markers"`, `"text"` joined with `"+"`, or one of `"none"`.

`name` `go_scattergeo()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_scattergeo_marker_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

`opacity` `go_scattergeo()`: Sets the opacity of the trace. Number in the interval [0, 1].

`go_scattergeo_marker()`: Sets the marker opacity. Number in the interval [0, 1], or a vector of these.

`go_scattergeo_selected_marker()`: Sets the marker opacity of selected points. Number in the interval [0, 1].

`go_scattergeo_unselected_marker()`: Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

`selected` Named list or `go_scattergeo_selected()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the 'unselected' are turned on for all points, whereas, any other non-array values means no selection all where the 'selected' and 'unselected' styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`text` `go_scattergeo()`: Sets text elements associated with each (lon,lat) pair or item in 'locations'. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (lon,lat) or 'locations' coordinates. If trace 'hoverinfo' contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_scattergeo_legendgrouptitle()`: Sets the title of the legend group. String.

`go_scattergeo_marker_colorbar_title()`: Sets the title of the color bar. String.

`textfont` `go_scattergeo()`: Sets the text font. Named list or `go_scattergeo_textfont()` object.

`go_scattergeo_selected()`: Named list or `go_scattergeo_selected_textfont()` object.

`go_scattergeo_unselected()`: Named list or `go_scattergeo_unselected_textfont()` object.

`textposition` Sets the positions of the 'text' elements with respects to the (x,y) coordinates. One of `"top left"`, `"top center"`, `"top right"`, `"middle left"`, `"middle center"`, `"middle right"`, `"bottom left"`, `"bottom center"`, `"bottom right"`, or a vector of these.

`texttemplate` Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'lat', 'lon', 'location' and 'text'. String, or a vector of these.

texttemplatefallback
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

uid
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision
: Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

unselected
: Named list or `go_scattergeo_unselected()` object.

visible
: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

row
: Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

col
: Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

skip_invalid
: If `TRUE`, silently drop invalid properties instead of raising an error.

arg
: Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

align
: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

bgcolor
: `go_scattergeo_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

  `go_scattergeo_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

bordercolor
: `go_scattergeo_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

  `go_scattergeo_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

font
: `go_scattergeo_hoverlabel()`: Sets the font used in hover labels. Named list or `go_scattergeo_hoverlabel_font()` object.

  `go_scattergeo_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_scattergeo_legendgrouptitle_font()` object.

  `go_scattergeo_marker_colorbar_title()`: Sets this color bar's title font. Named list or `go_scattergeo_marker_colorbar_title_font()` object.

namelength — Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to ‘namelength - 3‘ characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow — Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color — go_scattergeo_hoverlabel_font(), go_scattergeo_textfont(): Color: a CSS color string or R color name, or a vector of these.

go_scattergeo_legendgrouptitle_font(), go_scattergeo_marker_colorbar_tickfont(), go_scattergeo_marker_colorbar_title_font(): Color: a CSS color string or R color name.

go_scattergeo_line(): Sets the line color. Color: a CSS color string or R color name.

go_scattergeo_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.cmin‘ and ‘marker.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scattergeo.marker.colorscale, or a vector of these.

go_scattergeo_marker_gradient(): Sets the final color of the gradient fill: the center color for radial, the right for horizontal, or the bottom for vertical. Color: a CSS color string or R color name, or a vector of these.

go_scattergeo_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scattergeo.marker.line.colorscale, or a vector of these.

go_scattergeo_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name.

go_scattergeo_selected_textfont(): Sets the text font color of selected points. Color: a CSS color string or R color name.

go_scattergeo_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

go_scattergeo_unselected_textfont(): Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

family — go_scattergeo_hoverlabel_font(), go_scattergeo_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String, or a vector of these.

go_scattergeo_legendgrouptitle_font(), go_scattergeo_marker_colorbar_tickfont(), go_scattergeo_marker_colorbar_title_font(): HTML font family - the

typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: `go_scattergeo_hoverlabel_font()`, `go_scattergeo_textfont()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

`go_scattergeo_legendgrouptitle_font()`, `go_scattergeo_marker_colorbar_tickfont()`, `go_scattergeo_marker_colorbar_title_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

shadow
: `go_scattergeo_hoverlabel_font()`, `go_scattergeo_textfont()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

`go_scattergeo_legendgrouptitle_font()`, `go_scattergeo_marker_colorbar_tickfont()`, `go_scattergeo_marker_colorbar_title_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: `go_scattergeo_hoverlabel_font()`, `go_scattergeo_textfont()`: Number in the interval [1, Inf], or a vector of these.

`go_scattergeo_legendgrouptitle_font()`, `go_scattergeo_marker_colorbar_tickfont()`, `go_scattergeo_marker_colorbar_title_font()`: Number in the interval [1, Inf].

`go_scattergeo_marker()`: Sets the marker size (in px). Number in the interval [0, Inf], or a vector of these.

`go_scattergeo_selected_marker()`: Sets the marker size of selected points. Number in the interval [0, Inf].

`go_scattergeo_unselected_marker()`: Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

style
: `go_scattergeo_hoverlabel_font()`, `go_scattergeo_textfont()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`, or a vector of these.

`go_scattergeo_legendgrouptitle_font()`, `go_scattergeo_marker_colorbar_tickfont()`, `go_scattergeo_marker_colorbar_title_font()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

textcase
: `go_scattergeo_hoverlabel_font()`, `go_scattergeo_textfont()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`, or a vector of these.

`go_scattergeo_legendgrouptitle_font()`, `go_scattergeo_marker_colorbar_tickfont()`, `go_scattergeo_marker_colorbar_title_font()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.

variant — go_scattergeo_hoverlabel_font(), go_scattergeo_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_scattergeo_legendgrouptitle_font(), go_scattergeo_marker_colorbar_tickfont(), go_scattergeo_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight — go_scattergeo_hoverlabel_font(), go_scattergeo_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_scattergeo_legendgrouptitle_font(), go_scattergeo_marker_colorbar_tickfont(), go_scattergeo_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

dash — go_scattergeo_line(): Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\

go_scattergeo_marker_line(): Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\ or a vector of these.

width — go_scattergeo_line(): Sets the line width (in px). Number in the interval [0, Inf].

go_scattergeo_marker_line(): Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

angle — Sets the marker angle in respect to ‘angleref‘. Angle in degrees, or a vector of these.

angleref — Sets the reference for marker angle. With "previous", angle 0 points along the line from the previous point to this one. With "up", angle 0 points toward the top of the screen. With "north", angle 0 points north based on the current map projection. One of "previous", "up", "north".

autocolorscale — go_scattergeo_marker(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

go_scattergeo_marker_line(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

cauto — go_scattergeo_marker(): Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set

in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

`go_scattergeo_marker_line()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

`cmax` `go_scattergeo_marker()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

`go_scattergeo_marker_line()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

`cmid` `go_scattergeo_marker()`: Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

`go_scattergeo_marker_line()`: Sets the mid-point of the color domain by scaling ‘marker.line.cmin‘ and/or ‘marker.line.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

`cmin` `go_scattergeo_marker()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`go_scattergeo_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_scattergeo_marker_colorbar()` object.

`colorscale` `go_scattergeo_marker()`: Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list:

Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Re
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

go_scattergeo_marker_line(): Sets the colorscale. Has an effect only if in 'marker.line.color' is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the bounds of the colorscale in color space, use 'marker.line.cmin' and 'marker.line.cmax'. Alternatively, 'colorscale' may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,H
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

gradient — Named list or go_scattergeo_marker_gradient() object.

reversescale — go_scattergeo_marker(): Reverses the color mapping if true. Has an effect only if in 'marker.color' is set to a numerical array. If true, 'marker.cmin' will correspond to the last color in the array and 'marker.cmax' will correspond to the first color. Logical.

go_scattergeo_marker_line(): Reverses the color mapping if true. Has an effect only if in 'marker.line.color' is set to a numerical array. If true, 'marker.line.cmin' will correspond to the last color in the array and 'marker.line.cmax' will correspond to the first color. Logical.

showscale — Determines whether or not a colorbar is displayed for this trace. Has an effect only if in 'marker.color' is set to a numerical array. Logical.

sizemin — Has an effect only if 'marker.size' is set to a numerical array. Sets the minimum size (in px) of the rendered marker points. Number in the interval [0, Inf].

sizemode — Has an effect only if 'marker.size' is set to a numerical array. Sets the rule for which the data in 'size' is converted to pixels. One of "diameter", "area".

sizeref — Has an effect only if 'marker.size' is set to a numerical array. Sets the scale factor used to determine the rendered size of marker points. Use with 'sizemin' and 'sizemode'. Number.

standoff — Moves the marker away from the data point in the direction of 'angle' (in px). This can be useful for example if you have another marker at this location and you want to point an arrowhead marker at it. Number in the interval [0, Inf], or a vector of these.

symbol — Sets the marker symbol type. Adding 100 is equivalent to appending "-open" to a symbol name. Adding 200 is equivalent to appending "-dot" to a symbol name. Adding 300 is equivalent to appending "-open-dot" or "dot-open" to a symbol name. One of "0", "circle", "100", "circle-open", "200", "circle-dot", "300", "circle-open-dot", "1", "square", "101", "square-open", "201", "square-dot", "301", "square-open-dot", "2", "diamond", "102", "diamond-open", "202", "diamond-dot", "302", "diamond-open-dot", "3", "cross", "103", "cross-open", "203", "cross-dot", "303", "cross-open-dot", "4", "x", "104", "x-open", "204", "x-dot", "304", "x-open-dot", "5", "triangle-up",

"105", "triangle-up-open", "205", "triangle-up-dot", "305", "triangle-up-open-dot",
"6", "triangle-down", "106", "triangle-down-open", "206", "triangle-down-dot",
"306", "triangle-down-open-dot", "7", "triangle-left", "107", "triangle-left-open",
"207", "triangle-left-dot", "307", "triangle-left-open-dot", "8", "triangle-right",
"108", "triangle-right-open", "208", "triangle-right-dot", "308", "triangle-right-open-do
"9", "triangle-ne", "109", "triangle-ne-open", "209", "triangle-ne-dot",
"309", "triangle-ne-open-dot", "10", "triangle-se", "110", "triangle-se-open",
"210", "triangle-se-dot", "310", "triangle-se-open-dot", "11", "triangle-sw",
"111", "triangle-sw-open", "211", "triangle-sw-dot", "311", "triangle-sw-open-dot",
"12", "triangle-nw", "112", "triangle-nw-open", "212", "triangle-nw-dot",
"312", "triangle-nw-open-dot", "13", "pentagon", "113", "pentagon-open",
"213", "pentagon-dot", "313", "pentagon-open-dot", "14", "hexagon",
"114", "hexagon-open", "214", "hexagon-dot", "314", "hexagon-open-dot",
"15", "hexagon2", "115", "hexagon2-open", "215", "hexagon2-dot", "315",
"hexagon2-open-dot", "16", "octagon", "116", "octagon-open", "216",
"octagon-dot", "316", "octagon-open-dot", "17", "star", "117", "star-open",
"217", "star-dot", "317", "star-open-dot", "18", "hexagram", "118", "hexagram-open",
"218", "hexagram-dot", "318", "hexagram-open-dot", "19", "star-triangle-up",
"119", "star-triangle-up-open", "219", "star-triangle-up-dot", "319",
"star-triangle-up-open-dot", "20", "star-triangle-down", "120", "star-triangle-down-open
"220", "star-triangle-down-dot", "320", "star-triangle-down-open-dot",
"21", "star-square", "121", "star-square-open", "221", "star-square-dot",
"321", "star-square-open-dot", "22", "star-diamond", "122", "star-diamond-open",
"222", "star-diamond-dot", "322", "star-diamond-open-dot", "23", "diamond-tall",
"123", "diamond-tall-open", "223", "diamond-tall-dot", "323", "diamond-tall-open-dot",
"24", "diamond-wide", "124", "diamond-wide-open", "224", "diamond-wide-dot",
"324", "diamond-wide-open-dot", "25", "hourglass", "125", "hourglass-open",
"26", "bowtie", "126", "bowtie-open", "27", "circle-cross", "127", "circle-cross-open",
"28", "circle-x", "128", "circle-x-open", "29", "square-cross", "129",
"square-cross-open", "30", "square-x", "130", "square-x-open", "31",
"diamond-cross", "131", "diamond-cross-open", "32", "diamond-x", "132",
"diamond-x-open", "33", "cross-thin", "133", "cross-thin-open", "34",
"x-thin", "134", "x-thin-open", "35", "asterisk", "135", "asterisk-open",
"36", "hash", "136", "hash-open", "236", "hash-dot", "336", "hash-open-dot",
"37", "y-up", "137", "y-up-open", "38", "y-down", "138", "y-down-open",
"39", "y-left", "139", "y-left-open", "40", "y-right", "140", "y-right-open",
"41", "line-ew", "141", "line-ew-open", "42", "line-ns", "142", "line-ns-open",
"43", "line-ne", "143", "line-ne-open", "44", "line-nw", "144", "line-nw-open",
"45", "arrow-up", "145", "arrow-up-open", "46", "arrow-down", "146",
"arrow-down-open", "47", "arrow-left", "147", "arrow-left-open", "48",
"arrow-right", "148", "arrow-right-open", "49", "arrow-bar-up", "149",
"arrow-bar-up-open", "50", "arrow-bar-down", "150", "arrow-bar-down-open",
"51", "arrow-bar-left", "151", "arrow-bar-left-open", "52", "arrow-bar-right",
"152", "arrow-bar-right-open", "53", "arrow", "153", "arrow-open", "54",
"arrow-wide", "154", "arrow-wide-open", 0, 100, 200, 300, 1, 101, 201, 301,
2, 102, 202, 302, 3, 103, 203, 303, 4, 104, 204, 304, 5, 105, 205, 305, 6, 106,
206, 306, 7, 107, 207, 307, 8, 108, 208, 308, 9, 109, 209, 309, 10, 110, 210, 310,
11, 111, 211, 311, 12, 112, 212, 312, 13, 113, 213, 313, 14, 114, 214, 314, 15,

115, 215, 315, 16, 116, 216, 316, 17, 117, 217, 317, 18, 118, 218, 318, 19, 119, 219, 319, 20, 120, 220, 320, 21, 121, 221, 321, 22, 122, 222, 322, 23, 123, 223, 323, 24, 124, 224, 324, 25, 125, 26, 126, 27, 127, 28, 128, 29, 129, 30, 130, 31, 131, 32, 132, 33, 133, 34, 134, 35, 135, 36, 136, 236, 336, 37, 137, 38, 138, 39, 139, 40, 140, 41, 141, 42, 142, 43, 143, 44, 144, 45, 145, 46, 146, 47, 147, 48, 148, 49, 149, 50, 150, 51, 151, 52, 152, 53, 153, 54, 154, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick

is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".

showticksuffix Same as 'showtickprefix' but for tick suffixes. One of "all", "first", "last", "none".

thickness Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of "fraction", "pixels".

tick0 Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar's tick label font Named list or go_scattergeo_marker_colorbar_tickfont() object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or go_scattergeo_marker_colorbar_tickformatstop() objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_scattergeo_marker_colorbar_title()` object.

`x` Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`xanchor` Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`y` Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

yanchor — Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

side — Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of "right", "top", "bottom".

type — Sets the type of gradient used to fill the markers One of "radial", "horizontal", "vertical", "none", or a vector of these.

Details

`go_scattergeo()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_scattergeo()` creates an object and adds it to a figure.

Nested objects documented here:

`go_scattergeo_hoverlabel()` `scattergeo.hoverlabel`

`go_scattergeo_hoverlabel_font()` `scattergeo.hoverlabel.font`: Sets the font used in hover labels.

`go_scattergeo_legendgrouptitle()` `scattergeo.legendgrouptitle`

`go_scattergeo_legendgrouptitle_font()` `scattergeo.legendgrouptitle.font`: Sets this legend group's title font.

`go_scattergeo_line()` `scattergeo.line`

`go_scattergeo_marker()` `scattergeo.marker`

`go_scattergeo_marker_colorbar()` `scattergeo.marker.colorbar`

`go_scattergeo_marker_colorbar_tickfont()` `scattergeo.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_scattergeo_marker_colorbar_tickformatstop()` `scattergeo.marker.colorbar.tickformatstop`

`go_scattergeo_marker_colorbar_title()` `scattergeo.marker.colorbar.title`

`go_scattergeo_marker_colorbar_title_font()` `scattergeo.marker.colorbar.title.font`: Sets this color bar's title font.

`go_scattergeo_marker_gradient()` `scattergeo.marker.gradient`

`go_scattergeo_marker_line()` `scattergeo.marker.line`

`go_scattergeo_selected()` `scattergeo.selected`

`go_scattergeo_selected_marker()` `scattergeo.selected.marker`

`go_scattergeo_selected_textfont()` `scattergeo.selected.textfont`

`go_scattergeo_textfont()` `scattergeo.textfont`: Sets the text font.

`go_scattergeo_unselected()` `scattergeo.unselected`

`go_scattergeo_unselected_marker()` `scattergeo.unselected.marker`

`go_scattergeo_unselected_textfont()` `scattergeo.unselected.textfont`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_scattergeo(mode = "markers", marker_color = "red")
```

---

add_scattergl *Scattergl traces*

Description

The data visualized as scatter point or lines is set in 'x' and 'y' using the WebGL plotting engine. Bubble charts are achieved by setting 'marker.size' and/or 'marker.color' to a numerical arrays.

Usage

```
add_scattergl(fig, ...)

## Default S3 method:
add_scattergl(
  fig,
  ...,
  data = NULL,
  connectgaps = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  error_x = NULL,
  error_y = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
```

```
  xcalendar = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  yperiod = NULL,
  yperiod0 = NULL,
  yperiodalignment = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_scattergl(
  arg = NULL,
  ...,
  connectgaps = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  error_x = NULL,
  error_y = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
```

```
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  yperiod = NULL,
  yperiod0 = NULL,
  yperiodalignment = NULL,
  skip_invalid = FALSE
)

go_scattergl_error_x(
  arg = NULL,
  ...,
  array = NULL,
  arrayminus = NULL,
  color = NULL,
  copy_ystyle = NULL,
  symmetric = NULL,
  thickness = NULL,
  traceref = NULL,
  tracerefminus = NULL,
  type = NULL,
  value = NULL,
  valueminus = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)
```

```
go_scattergl_error_y(
  arg = NULL,
  ...,
  array = NULL,
  arrayminus = NULL,
  color = NULL,
  symmetric = NULL,
  thickness = NULL,
  traceref = NULL,
  tracerefminus = NULL,
  type = NULL,
  value = NULL,
  valueminus = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scattergl_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_scattergl_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattergl_legendgrouptitle(
  arg = NULL,
```

```
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scattergl_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattergl_line(
  arg = NULL,
  ...,
  color = NULL,
  dash = NULL,
  shape = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scattergl_marker(
  arg = NULL,
  ...,
  angle = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  line = NULL,
  opacity = NULL,
  reversescale = NULL,
  showscale = NULL,
```

```
  size = NULL,
  sizemin = NULL,
  sizemode = NULL,
  sizeref = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_scattergl_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
```

```
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_scattergl_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattergl_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_scattergl_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scattergl_marker_colorbar_title_font(
```

```
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattergl_marker_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scattergl_selected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scattergl_selected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scattergl_selected_textfont(
```

```
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

go_scattergl_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  size = NULL,
  style = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattergl_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scattergl_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scattergl_unselected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)
```

Arguments

`fig`	A `go_figure` object.
`...`	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.

`data` A data frame in which formula properties (e.g. `x = ~col`, `marker_size = ~col`) are evaluated. Default: the data given to `go_figure()`. Only the evaluated values are added to the trace. A `crosstalk::SharedData` works the same way, and links the trace to its crosstalk group (see `highlight()`).

`connectgaps` Determines whether or not gaps (i.e. {nan} or missing values) in the provided data arrays are connected. Logical.

`customdata` Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`dx` Sets the x coordinate step. See ‘x0‘ for more info. Number.

`dy` Sets the y coordinate step. See ‘y0‘ for more info. Number.

`error_x` Named list or `go_scattergl_error_x()` object.

`error_y` Named list or `go_scattergl_error_y()` object.

`fill` Sets the area to fill with a solid color. Defaults to "none" unless this trace is stacked, then it gets "tonexty" ("tonextx") if ‘orientation‘ is "v" ("h") Use with ‘fillcolor‘ if not "none". "tozerox" and "tozeroy" fill to x=0 and y=0 respectively. "tonextx" and "tonexty" fill between the endpoints of this trace and the endpoints of the trace before it, connecting those endpoints with straight lines (to make a stacked area graph); if there is no trace before it, they behave like "tozerox" and "tozeroy". "toself" connects the endpoints of the trace (or each segment of the trace if it has gaps) into a closed shape. "tonext" fills the space between two traces if one completely encloses the other (eg consecutive contour lines), and behaves like "toself" if there is no trace before it. "tonext" should not be used if one trace does not enclose the other. Traces in a ‘stackgroup‘ will only fill to (or be filled to) other traces in the same group. With multiple ‘stackgroup‘s or some traces stacked and some not, if fill-linked traces are not already consecutive, the later ones will be pushed down in the drawing order. One of `"none"`, `"tozeroy"`, `"tozerox"`, `"tonexty"`, `"tonextx"`, `"toself"`, `"tonext"`.

`fillcolor` Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name.

`hoverinfo` Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_scattergl_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for

example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_scattergl_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line`
: `go_scattergl()`: Named list or `go_scattergl_line()` object.

`go_scattergl_marker()`: Named list or `go_scattergl_marker_line()` object.

marker
: go_scattergl(): Named list or go_scattergl_marker() object.
go_scattergl_selected(): Named list or go_scattergl_selected_marker() object.
go_scattergl_unselected(): Named list or go_scattergl_unselected_marker() object.

meta
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

mode
: Determines the drawing mode for this scatter trace. Any combination of "lines", "markers", "text" joined with "+", or one of "none".

name
: go_scattergl(): Sets the trace name. The trace name appears as the legend item and on hover. String.
go_scattergl_marker_colorbar_tickformatstop(): When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

opacity
: go_scattergl(): Sets the opacity of the trace. Number in the interval [0, 1].
go_scattergl_marker(): Sets the marker opacity. Number in the interval [0, 1], or a vector of these.
go_scattergl_selected_marker(): Sets the marker opacity of selected points. Number in the interval [0, 1].
go_scattergl_unselected_marker(): Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

selected
: Named list or go_scattergl_selected() object.

selectedpoints
: Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.

showlegend
: Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

text
: go_scattergl(): Sets text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace’s (x,y) coordinates. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.
go_scattergl_legendgrouptitle(): Sets the title of the legend group. String.
go_scattergl_marker_colorbar_title(): Sets the title of the color bar. String.

textfont
: go_scattergl(): Sets the text font. Named list or go_scattergl_textfont() object.
go_scattergl_selected(): Named list or go_scattergl_selected_textfont() object.
go_scattergl_unselected(): Named list or go_scattergl_unselected_textfont() object.

textposition
: Sets the positions of the 'text' elements with respects to the (x,y) coordinates. One of "top left", "top center", "top right", "middle left", "middle center", "middle right", "bottom left", "bottom center", "bottom right", or a vector of these.

texttemplate
: Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. String, or a vector of these.

texttemplatefallback
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

uid
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision
: Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

unselected
: Named list or go_scattergl_unselected() object.

visible
: go_scattergl(): Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.
go_scattergl_error_x(), go_scattergl_error_y(): Determines whether or not this set of error bars is visible. Logical.

x — `go_scattergl()`: Sets the x coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_scattergl_marker_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

x0 — Alternate to ‘x‘. Builds a linear space of x coordinates. Use with ‘dx‘ where ‘x0‘ is the starting coordinate and ‘dx‘ the step. Any value.

xaxis — Sets a reference between this trace’s x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

xcalendar — Sets the calendar system to use with ‘x‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

xhoverformat — Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

xperiod — Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the x axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

xperiod0 — Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the x0 axis. When ‘x0period‘ is round number of weeks, the ‘x0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

xperiodalignment — Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the x axis. One of `"start"`, `"middle"`, `"end"`.

y — `go_scattergl()`: Sets the y coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_scattergl_marker_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

y0 — Alternate to ‘y‘. Builds a linear space of y coordinates. Use with ‘dy‘ where ‘y0‘ is the starting coordinate and ‘dy‘ the step. Any value.

`yaxis` Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to 'layout.yaxis'. If "y2", the y coordinates refer to 'layout.yaxis2', and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...

`ycalendar` Sets the calendar system to use with 'y' date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

`yhoverformat` Sets the hover text formatting rule for 'y' using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using 'yaxis.hoverformat'. String.

`yperiod` Only relevant when the axis 'type' is "date". Sets the period positioning in milliseconds or "M<n>" on the y axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case 'n' must be a positive integer. Any value.

`yperiod0` Only relevant when the axis 'type' is "date". Sets the base for period positioning in milliseconds or date string on the y0 axis. When 'y0period' is round number of weeks, the 'y0period0' by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

`yperiodalignment`
Only relevant when the axis 'type' is "date". Sets the alignment of data points on the y axis. One of `"start"`, `"middle"`, `"end"`.

`row` Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`col` Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`secondary_y` If `TRUE`, use the secondary y axis of the subplot.

`skip_invalid` If `TRUE`, silently drop invalid properties instead of raising an error.

`arg` Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

`array` Sets the data corresponding the length of each error bar. Values are plotted relative to the underlying data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`arrayminus` Sets the data corresponding the length of each error bar in the bottom (left) direction for vertical (horizontal) bars Values are plotted relative to the underlying data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`color` `go_scattergl_error_x()`, `go_scattergl_error_y()`: Sets the stroke color of the error bars. Color: a CSS color string or R color name.

`go_scattergl_hoverlabel_font()`, `go_scattergl_textfont()`: Color: a CSS color string or R color name, or a vector of these.

go_scattergl_legendgrouptitle_font(), go_scattergl_marker_colorbar_tickfont(), go_scattergl_marker_colorbar_title_font(): Color: a CSS color string or R color name.

go_scattergl_line(): Sets the line color. Color: a CSS color string or R color name.

go_scattergl_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.cmin' and 'marker.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by scattergl.marker.colorscale, or a vector of these.

go_scattergl_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.line.cmin' and 'marker.line.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by scattergl.marker.line.colorscale, or a vector of these.

go_scattergl_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name.

go_scattergl_selected_textfont(): Sets the text font color of selected points. Color: a CSS color string or R color name.

go_scattergl_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

go_scattergl_unselected_textfont(): Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

copy_ystyle Logical.

symmetric Determines whether or not the error bars have the same length in both direction (top/bottom for vertical bars, left/right for horizontal bars. Logical.

thickness go_scattergl_error_x(), go_scattergl_error_y(): Sets the thickness (in px) of the error bars. Number in the interval [0, Inf].

go_scattergl_marker_colorbar(): Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

traceref Integer in the interval [0, Inf].

tracerefminus Integer in the interval [0, Inf].

type Determines the rule used to generate the error bars. If "constant", the bar lengths are of a constant value. Set this constant in 'value'. If "percent", the bar lengths correspond to a percentage of underlying data. Set this percentage in 'value'. If "sqrt", the bar lengths correspond to the square of the underlying data. If "data", the bar lengths are set with data set 'array'. One of "percent", "constant", "sqrt", "data".

value go_scattergl_error_x(), go_scattergl_error_y(): Sets the value of either the percentage (if 'type' is set to "percent") or the constant (if 'type' is set to "constant") corresponding to the lengths of the error bars. Number in the interval [0, Inf].

	`go_scattergl_marker_colorbar_tickformatstop()`: string - dtickformat for described zoom level, the same as "tickformat" String.
`valueminus`	Sets the value of either the percentage (if 'type' is set to "percent") or the constant (if 'type' is set to "constant") corresponding to the lengths of the error bars in the bottom (left) direction for vertical (horizontal) bars Number in the interval [0, Inf].
`width`	`go_scattergl_error_x()`, `go_scattergl_error_y()`: Sets the width (in px) of the cross-bar at both ends of the error bars. Number in the interval [0, Inf]. `go_scattergl_line()`: Sets the line width (in px). Number in the interval [0, Inf]. `go_scattergl_marker_line()`: Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.
`align`	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.
`bgcolor`	`go_scattergl_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these. `go_scattergl_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.
`bordercolor`	`go_scattergl_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these. `go_scattergl_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.
`font`	`go_scattergl_hoverlabel()`: Sets the font used in hover labels. Named list or `go_scattergl_hoverlabel_font()` object. `go_scattergl_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_scattergl_legendgrouptitle_font()` object. `go_scattergl_marker_colorbar_title()`: Sets this color bar's title font. Named list or `go_scattergl_marker_colorbar_title_font()` object.
`namelength`	Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.
`showarrow`	Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.
`family`	`go_scattergl_hoverlabel_font()`, `go_scattergl_textfont()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these. `go_scattergl_legendgrouptitle_font()`, `go_scattergl_marker_colorbar_tickfont()`, `go_scattergl_marker_colorbar_title_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only

apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: `go_scattergl_hoverlabel_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

`go_scattergl_legendgrouptitle_font()`, `go_scattergl_marker_colorbar_tickfont()`, `go_scattergl_marker_colorbar_title_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

shadow
: `go_scattergl_hoverlabel_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

`go_scattergl_legendgrouptitle_font()`, `go_scattergl_marker_colorbar_tickfont()`, `go_scattergl_marker_colorbar_title_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: `go_scattergl_hoverlabel_font()`, `go_scattergl_textfont()`: Number in the interval [1, Inf], or a vector of these.

`go_scattergl_legendgrouptitle_font()`, `go_scattergl_marker_colorbar_tickfont()`, `go_scattergl_marker_colorbar_title_font()`: Number in the interval [1, Inf].

`go_scattergl_marker()`: Sets the marker size (in px). Number in the interval [0, Inf], or a vector of these.

`go_scattergl_selected_marker()`: Sets the marker size of selected points. Number in the interval [0, Inf].

`go_scattergl_unselected_marker()`: Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

style
: `go_scattergl_hoverlabel_font()`, `go_scattergl_textfont()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`, or a vector of these.

`go_scattergl_legendgrouptitle_font()`, `go_scattergl_marker_colorbar_tickfont()`, `go_scattergl_marker_colorbar_title_font()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

textcase
: `go_scattergl_hoverlabel_font()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`, or a vector of these.

`go_scattergl_legendgrouptitle_font()`, `go_scattergl_marker_colorbar_tickfont()`, `go_scattergl_marker_colorbar_title_font()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.

variant
: go_scattergl_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.
go_scattergl_legendgrouptitle_font(), go_scattergl_marker_colorbar_tickfont(), go_scattergl_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
go_scattergl_textfont(): Sets the variant of the font. One of "normal", "small-caps", or a vector of these.

weight
: go_scattergl_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.
go_scattergl_legendgrouptitle_font(), go_scattergl_marker_colorbar_tickfont(), go_scattergl_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
go_scattergl_textfont(): Sets the weight (or boldness) of the font. One of "normal", "bold", or a vector of these.

dash
: Sets the style of the lines. One of "dash", "dashdot", "dot", "longdash", "longdashdot", "solid".

shape
: Determines the line shape. The values correspond to step-wise line shapes. One of "linear", "hv", "vh", "hvh", "vhv".

angle
: Sets the marker angle in respect to ‘angleref‘. Angle in degrees, or a vector of these.

autocolorscale
: go_scattergl_marker(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
go_scattergl_marker_line(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

cauto
: go_scattergl_marker(): Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.
go_scattergl_marker_line(): Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

cmax
: go_scattergl_marker(): Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have

the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

`go_scattergl_marker_line()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

`cmid` `go_scattergl_marker()`: Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

`go_scattergl_marker_line()`: Sets the mid-point of the color domain by scaling ‘marker.line.cmin‘ and/or ‘marker.line.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

`cmin` `go_scattergl_marker()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`go_scattergl_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_scattergl_marker_colorbar()` object.

`colorscale` `go_scattergl_marker()`: Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Re
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_scattergl_marker_line()`: Sets the colorscale. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.line.cmin‘ and ‘marker.line.cmax‘. Alternatively, ‘colorscale‘ may be a

palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,H
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`reversescale` `go_scattergl_marker()`: Reverses the color mapping if true. Has an effect only if in 'marker.color' is set to a numerical array. If true, 'marker.cmin' will correspond to the last color in the array and 'marker.cmax' will correspond to the first color. Logical.

`go_scattergl_marker_line()`: Reverses the color mapping if true. Has an effect only if in 'marker.line.color' is set to a numerical array. If true, 'marker.line.cmin' will correspond to the last color in the array and 'marker.line.cmax' will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if in 'marker.color' is set to a numerical array. Logical.

`sizemin` Has an effect only if 'marker.size' is set to a numerical array. Sets the minimum size (in px) of the rendered marker points. Number in the interval [0, Inf].

`sizemode` Has an effect only if 'marker.size' is set to a numerical array. Sets the rule for which the data in 'size' is converted to pixels. One of `"diameter"`, `"area"`.

`sizeref` Has an effect only if 'marker.size' is set to a numerical array. Sets the scale factor used to determine the rendered size of marker points. Use with 'sizemin' and 'sizemode'. Number.

`symbol` Sets the marker symbol type. Adding 100 is equivalent to appending "-open" to a symbol name. Adding 200 is equivalent to appending "-dot" to a symbol name. Adding 300 is equivalent to appending "-open-dot" or "dot-open" to a symbol name. One of `"0"`, `"circle"`, `"100"`, `"circle-open"`, `"200"`, `"circle-dot"`, `"300"`, `"circle-open-dot"`, `"1"`, `"square"`, `"101"`, `"square-open"`, `"201"`, `"square-dot"`, `"301"`, `"square-open-dot"`, `"2"`, `"diamond"`, `"102"`, `"diamond-open"`, `"202"`, `"diamond-dot"`, `"302"`, `"diamond-open-dot"`, `"3"`, `"cross"`, `"103"`, `"cross-open"`, `"203"`, `"cross-dot"`, `"303"`, `"cross-open-dot"`, `"4"`, `"x"`, `"104"`, `"x-open"`, `"204"`, `"x-dot"`, `"304"`, `"x-open-dot"`, `"5"`, `"triangle-up"`, `"105"`, `"triangle-up-open"`, `"205"`, `"triangle-up-dot"`, `"305"`, `"triangle-up-open-dot"`, `"6"`, `"triangle-down"`, `"106"`, `"triangle-down-open"`, `"206"`, `"triangle-down-dot"`, `"306"`, `"triangle-down-open-dot"`, `"7"`, `"triangle-left"`, `"107"`, `"triangle-left-open"`, `"207"`, `"triangle-left-dot"`, `"307"`, `"triangle-left-open-dot"`, `"8"`, `"triangle-right"`, `"108"`, `"triangle-right-open"`, `"208"`, `"triangle-right-dot"`, `"308"`, `"triangle-right-open-do`
`"9"`, `"triangle-ne"`, `"109"`, `"triangle-ne-open"`, `"209"`, `"triangle-ne-dot"`, `"309"`, `"triangle-ne-open-dot"`, `"10"`, `"triangle-se"`, `"110"`, `"triangle-se-open"`, `"210"`, `"triangle-se-dot"`, `"310"`, `"triangle-se-open-dot"`, `"11"`, `"triangle-sw"`, `"111"`, `"triangle-sw-open"`, `"211"`, `"triangle-sw-dot"`, `"311"`, `"triangle-sw-open-dot"`, `"12"`, `"triangle-nw"`, `"112"`, `"triangle-nw-open"`, `"212"`, `"triangle-nw-dot"`, `"312"`, `"triangle-nw-open-dot"`, `"13"`, `"pentagon"`, `"113"`, `"pentagon-open"`, `"213"`, `"pentagon-dot"`, `"313"`, `"pentagon-open-dot"`, `"14"`, `"hexagon"`, `"114"`, `"hexagon-open"`, `"214"`, `"hexagon-dot"`, `"314"`, `"hexagon-open-dot"`, `"15"`, `"hexagon2"`, `"115"`, `"hexagon2-open"`, `"215"`, `"hexagon2-dot"`, `"315"`, `"hexagon2-open-dot"`, `"16"`, `"octagon"`, `"116"`, `"octagon-open"`, `"216"`, `"octagon-dot"`, `"316"`, `"octagon-open-dot"`, `"17"`, `"star"`, `"117"`, `"star-open"`, `"217"`, `"star-dot"`, `"317"`, `"star-open-dot"`, `"18"`, `"hexagram"`, `"118"`, `"hexagram-open"`,

"218", "hexagram-dot", "318", "hexagram-open-dot", "19", "star-triangle-up",
"119", "star-triangle-up-open", "219", "star-triangle-up-dot", "319",
"star-triangle-up-open-dot", "20", "star-triangle-down", "120", "star-triangle-down-open
"220", "star-triangle-down-dot", "320", "star-triangle-down-open-dot",
"21", "star-square", "121", "star-square-open", "221", "star-square-dot",
"321", "star-square-open-dot", "22", "star-diamond", "122", "star-diamond-open",
"222", "star-diamond-dot", "322", "star-diamond-open-dot", "23", "diamond-tall",
"123", "diamond-tall-open", "223", "diamond-tall-dot", "323", "diamond-tall-open-dot",
"24", "diamond-wide", "124", "diamond-wide-open", "224", "diamond-wide-dot",
"324", "diamond-wide-open-dot", "25", "hourglass", "125", "hourglass-open",
"26", "bowtie", "126", "bowtie-open", "27", "circle-cross", "127", "circle-cross-open",
"28", "circle-x", "128", "circle-x-open", "29", "square-cross", "129",
"square-cross-open", "30", "square-x", "130", "square-x-open", "31",
"diamond-cross", "131", "diamond-cross-open", "32", "diamond-x", "132",
"diamond-x-open", "33", "cross-thin", "133", "cross-thin-open", "34",
"x-thin", "134", "x-thin-open", "35", "asterisk", "135", "asterisk-open",
"36", "hash", "136", "hash-open", "236", "hash-dot", "336", "hash-open-dot",
"37", "y-up", "137", "y-up-open", "38", "y-down", "138", "y-down-open",
"39", "y-left", "139", "y-left-open", "40", "y-right", "140", "y-right-open",
"41", "line-ew", "141", "line-ew-open", "42", "line-ns", "142", "line-ns-open",
"43", "line-ne", "143", "line-ne-open", "44", "line-nw", "144", "line-nw-open",
"45", "arrow-up", "145", "arrow-up-open", "46", "arrow-down", "146",
"arrow-down-open", "47", "arrow-left", "147", "arrow-left-open", "48",
"arrow-right", "148", "arrow-right-open", "49", "arrow-bar-up", "149",
"arrow-bar-up-open", "50", "arrow-bar-down", "150", "arrow-bar-down-open",
"51", "arrow-bar-left", "151", "arrow-bar-left-open", "52", "arrow-bar-right",
"152", "arrow-bar-right-open", "53", "arrow", "153", "arrow-open", "54",
"arrow-wide", "154", "arrow-wide-open", 0, 100, 200, 300, 1, 101, 201, 301,
2, 102, 202, 302, 3, 103, 203, 303, 4, 104, 204, 304, 5, 105, 205, 305, 6, 106,
206, 306, 7, 107, 207, 307, 8, 108, 208, 308, 9, 109, 209, 309, 10, 110, 210, 310,
11, 111, 211, 311, 12, 112, 212, 312, 13, 113, 213, 313, 14, 114, 214, 314, 15,
115, 215, 315, 16, 116, 216, 316, 17, 117, 217, 317, 18, 118, 218, 318, 19, 119,
219, 319, 20, 120, 220, 320, 21, 121, 221, 321, 22, 122, 222, 322, 23, 123, 223,
323, 24, 124, 224, 324, 25, 125, 26, 126, 27, 127, 28, 128, 29, 129, 30, 130, 31,
131, 32, 132, 33, 133, 34, 134, 35, 135, 36, 136, 236, 336, 37, 137, 38, 138, 39,
139, 40, 140, 41, 141, 42, 142, 43, 143, 44, 144, 45, 145, 46, 146, 47, 147, 48,
148, 49, 149, 50, 150, 51, 151, 52, 152, 53, 153, 54, 154, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30).

If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

labelalias Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

len Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

lenmode Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of "fraction", "pixels".

minexponent Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

nticks Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

orientation Sets the orientation of the colorbar. One of "h", "v".

outlinecolor Sets the axis line color. Color: a CSS color string or R color name.

outlinewidth Sets the width (in px) of the axis line. Number in the interval [0, Inf].

separatethousands
If "true", even 4-digit integers are separated Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".

showticksuffix Same as 'showtickprefix' but for tick suffixes. One of "all", "first", "last", "none".

thicknessmode Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of "fraction", "pixels".

tick0 Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where

each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_scattergl_marker_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops` List of named lists or `go_scattergl_marker_colorbar_tickformatstop()` objects.

`ticklabeloverflow` Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition` Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_scattergl_marker_colorbar_title()` object.

`xanchor` Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`yanchor` Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

`ypad` Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

`yref` Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

`dtickrange` range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

`enabled` Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

`templateitemname` Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`side` Determines the location of color bar’s title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.

Details

`go_scattergl()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_scattergl()` creates an object and adds it to a figure.

Nested objects documented here:

`go_scattergl_error_x()` `scattergl.error_x`

`go_scattergl_error_y()` `scattergl.error_y`

`go_scattergl_hoverlabel()` `scattergl.hoverlabel`

`go_scattergl_hoverlabel_font()` `scattergl.hoverlabel.font`: Sets the font used in hover labels.

`go_scattergl_legendgrouptitle()` `scattergl.legendgrouptitle`

`go_scattergl_legendgrouptitle_font()` `scattergl.legendgrouptitle.font`: Sets this legend group's title font.

`go_scattergl_line()` `scattergl.line`

`go_scattergl_marker()` `scattergl.marker`

`go_scattergl_marker_colorbar()` `scattergl.marker.colorbar`

`go_scattergl_marker_colorbar_tickfont()` `scattergl.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_scattergl_marker_colorbar_tickformatstop()` `scattergl.marker.colorbar.tickformatstop`

`go_scattergl_marker_colorbar_title()` `scattergl.marker.colorbar.title`

`go_scattergl_marker_colorbar_title_font()` `scattergl.marker.colorbar.title.font`: Sets this color bar's title font.

`go_scattergl_marker_line()` `scattergl.marker.line`

`go_scattergl_selected()` `scattergl.selected`

`go_scattergl_selected_marker()` `scattergl.selected.marker`

`go_scattergl_selected_textfont()` `scattergl.selected.textfont`

`go_scattergl_textfont()` `scattergl.textfont`: Sets the text font.

`go_scattergl_unselected()` `scattergl.unselected`

`go_scattergl_unselected_marker()` `scattergl.unselected.marker`

`go_scattergl_unselected_textfont()` `scattergl.unselected.textfont`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_scattergl(x = 1:3, y = c(2, 1, 3), mode = "markers", marker_color = "red")
```

add_scattermap *Scattermap traces*

Description

The data visualized as scatter point, lines or marker symbols on a MapLibre GL geographic map is provided by longitude/latitude pairs in ‘lon‘ and ‘lat‘.

Usage

```
add_scattermap(fig, ...)

## Default S3 method:
add_scattermap(
  fig,
  ...,
  data = NULL,
  below = NULL,
  cluster = NULL,
  connectgaps = NULL,
  customdata = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  lat = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  lon = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  subplot = NULL,
  text = NULL,
```

```
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_scattermap(
  arg = NULL,
  ...,
  below = NULL,
  cluster = NULL,
  connectgaps = NULL,
  customdata = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  lat = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  lon = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  subplot = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
```

```
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_scattermap_cluster(
  arg = NULL,
  ...,
  color = NULL,
  enabled = NULL,
  maxzoom = NULL,
  opacity = NULL,
  size = NULL,
  step = NULL,
  skip_invalid = FALSE
)

go_scattermap_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_scattermap_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

```
  go_scattermap_legendgrouptitle(
    arg = NULL,
    ...,
    font = NULL,
    text = NULL,
    skip_invalid = FALSE
  )

  go_scattermap_legendgrouptitle_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_scattermap_line(
    arg = NULL,
    ...,
    color = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_scattermap_marker(
    arg = NULL,
    ...,
    allowoverlap = NULL,
    angle = NULL,
    autocolorscale = NULL,
    cauto = NULL,
    cmax = NULL,
    cmid = NULL,
    cmin = NULL,
    color = NULL,
    coloraxis = NULL,
    colorbar = NULL,
    colorscale = NULL,
    opacity = NULL,
    reversescale = NULL,
    showscale = NULL,
```

```
  size = NULL,
  sizemin = NULL,
  sizemode = NULL,
  sizeref = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_scattermap_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
```

```
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_scattermap_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattermap_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_scattermap_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scattermap_marker_colorbar_title_font(
```

```
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattermap_selected(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

go_scattermap_selected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scattermap_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  size = NULL,
  style = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattermap_unselected(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

go_scattermap_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
below	Determines if this scattermap trace's layers are to be inserted before the layer with the specified ID. By default, scattermap layers are inserted above all the base layers. To place the scattermap layers above every other layer, set 'below' to "". String.
cluster	Named list or go_scattermap_cluster() object.
connectgaps	Determines whether or not gaps (i.e. {nan} or missing values) in the provided data arrays are connected. Logical.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
fill	Sets the area to fill with a solid color. Use with 'fillcolor' if not "none". "toself" connects the endpoints of the trace (or each segment of the trace if it has gaps) into a closed shape. One of "none", "toself".
fillcolor	Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name.
hoverinfo	Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of "lon", "lat", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_scattermap_hoverlabel() object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables

with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example ‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each (lon,lat) pair If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (lon,lat) coordinates. To be seen, trace ‘hoverinfo‘ must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`lat`
: Sets the latitude coordinates (in degrees North). Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_scattermap_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line`
: Named list or `go_scattermap_line()` object.

`lon`
: Sets the longitude coordinates (in degrees East). Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`marker`
: `go_scattermap()`: Named list or `go_scattermap_marker()` object.

`go_scattermap_selected()`: Named list or `go_scattermap_selected_marker()` object.

`go_scattermap_unselected()`: Named list or `go_scattermap_unselected_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`mode` Determines the drawing mode for this scatter trace. If the provided ‘mode‘ includes "text" then the ‘text‘ elements appear at the coordinates. Otherwise, the ‘text‘ elements appear on hover. Any combination of `"lines"`, `"markers"`, `"text"` joined with `"+"`, or one of `"none"`.

`name` `go_scattermap()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_scattermap_marker_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` `go_scattermap()`: Sets the opacity of the trace. Number in the interval [0, 1].

`go_scattermap_cluster()`, `go_scattermap_marker()`: Sets the marker opacity. Number in the interval [0, 1], or a vector of these.

`go_scattermap_selected_marker()`: Sets the marker opacity of selected points. Number in the interval [0, 1].

`go_scattermap_unselected_marker()`: Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

`selected` Named list or `go_scattermap_selected()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`subplot` Sets a reference between this trace’s data coordinates and a map subplot. If "map" (the default value), the data refer to ‘layout.map‘. If "map2", the data refer to ‘layout.map2‘, and so on. Subplot id: `"map"`, `"map2"`, `"map3"`, ...

`text` `go_scattermap()`: Sets text elements associated with each (lon,lat) pair If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace’s (lon,lat) coordinates. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_scattermap_legendgrouptitle()`: Sets the title of the legend group. String.

`go_scattermap_marker_colorbar_title()`: Sets the title of the color bar. String.

`textfont` Sets the icon text font (color=map.layer.paint.text-color, size=map.layer.layout.text-size). Has an effect only when 'type' is set to "symbol". Named list or `go_scattermap_textfont()` object.

`textposition` Sets the positions of the 'text' elements with respects to the (x,y) coordinates. One of `"top left"`, `"top center"`, `"top right"`, `"middle left"`, `"middle center"`, `"middle right"`, `"bottom left"`, `"bottom center"`, `"bottom right"`.

`texttemplate` Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'lat', 'lon' and 'text'. String, or a vector of these.

`texttemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

`unselected` Named list or `go_scattermap_unselected()` object.

`visible` Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`row` Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`col` Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`skip_invalid` If `TRUE`, silently drop invalid properties instead of raising an error.

arg — Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

color — go_scattermap_cluster(): Sets the color for each cluster step. Color: a CSS color string or R color name, or a vector of these.

go_scattermap_hoverlabel_font(): Color: a CSS color string or R color name, or a vector of these.

go_scattermap_legendgrouptitle_font(), go_scattermap_marker_colorbar_tickfont(), go_scattermap_marker_colorbar_title_font(), go_scattermap_textfont(): Color: a CSS color string or R color name.

go_scattermap_line(): Sets the line color. Color: a CSS color string or R color name.

go_scattermap_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.cmin' and 'marker.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by scattermap.marker.colorscale, or a vector of these.

go_scattermap_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name.

go_scattermap_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

enabled — go_scattermap_cluster(): Determines whether clustering is enabled or disabled. Logical.

go_scattermap_marker_colorbar_tickformatstop(): Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

maxzoom — Sets the maximum zoom level. At zoom levels equal to or greater than this, points will never be clustered. Number in the interval [0, 24].

size — go_scattermap_cluster(): Sets the size for each cluster step. Number in the interval [0, Inf], or a vector of these.

go_scattermap_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

go_scattermap_legendgrouptitle_font(), go_scattermap_marker_colorbar_tickfont(), go_scattermap_marker_colorbar_title_font(), go_scattermap_textfont(): Number in the interval [1, Inf].

go_scattermap_marker(): Sets the marker size (in px). Number in the interval [0, Inf], or a vector of these.

go_scattermap_selected_marker(): Sets the marker size of selected points. Number in the interval [0, Inf].

go_scattermap_unselected_marker(): Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

step — Sets how many points it takes to create a cluster or advance to the next cluster step. Use this in conjunction with arrays for 'size' and / or 'color'. If an integer, steps start at multiples of this number. If an array, each step extends from the given value until one less than the next value. Number in the interval [-1, Inf], or a vector of these.

`align` Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`bgcolor` `go_scattermap_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`go_scattermap_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

`bordercolor` `go_scattermap_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`go_scattermap_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

`font` `go_scattermap_hoverlabel()`: Sets the font used in hover labels. Named list or `go_scattermap_hoverlabel_font()` object.

`go_scattermap_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_scattermap_legendgrouptitle_font()` object.

`go_scattermap_marker_colorbar_title()`: Sets this color bar's title font. Named list or `go_scattermap_marker_colorbar_title_font()` object.

`namelength` Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

`showarrow` Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

`family` `go_scattermap_hoverlabel_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

`go_scattermap_legendgrouptitle_font()`, `go_scattermap_marker_colorbar_tickfont()`, `go_scattermap_marker_colorbar_title_font()`, `go_scattermap_textfont()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

`lineposition` `go_scattermap_hoverlabel_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

`go_scattermap_legendgrouptitle_font()`, `go_scattermap_marker_colorbar_tickfont()`, `go_scattermap_marker_colorbar_title_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

shadow
: go_scattermap_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.
go_scattermap_legendgrouptitle_font(), go_scattermap_marker_colorbar_tickfont(), go_scattermap_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

style
: go_scattermap_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.
go_scattermap_legendgrouptitle_font(), go_scattermap_marker_colorbar_tickfont(), go_scattermap_marker_colorbar_title_font(), go_scattermap_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_scattermap_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.
go_scattermap_legendgrouptitle_font(), go_scattermap_marker_colorbar_tickfont(), go_scattermap_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_scattermap_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.
go_scattermap_legendgrouptitle_font(), go_scattermap_marker_colorbar_tickfont(), go_scattermap_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_scattermap_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.
go_scattermap_legendgrouptitle_font(), go_scattermap_marker_colorbar_tickfont(), go_scattermap_marker_colorbar_title_font(), go_scattermap_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

width
: Sets the line width (in px). Number in the interval [0, Inf].

allowoverlap
: Flag to draw all symbols, even if they overlap. Logical.

angle
: Sets the marker orientation from true North, in degrees clockwise. When using the "auto" default, no rotation would be applied in perspective views which is different from using a zero angle. Number, or a vector of these.

autocolorscale
: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`cauto` Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

`cmax` Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

`cmid` Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

`cmin` Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_scattermap_marker_colorbar()` object.

`colorscale` Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`reversescale` Reverses the color mapping if true. Has an effect only if in ‘marker.color‘ is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘marker.color‘ is set to a numerical array. Logical.

`sizemin` Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the minimum size (in px) of the rendered marker points. Number in the interval [0, Inf].

`sizemode` Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the rule for which the data in ‘size‘ is converted to pixels. One of `"diameter"`, `"area"`.

`sizeref` Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the scale factor used to determine the rendered size of marker points. Use with ‘sizemin‘ and ‘sizemode‘. Number.

`symbol` Sets the marker symbol. Full list: https://www.mapbox.com/maki-icons/ Note that the array ‘marker.color‘ and ‘marker.size‘ are only available for "circle" symbols. String, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticksuffix Same as ‘showtickprefix‘ but for tick suffixes. One of "all", "first", "last", "none".

thickness Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of "fraction", "pixels".

tick0 Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar's tick label font Named list or go_scattermap_marker_colorbar_tickfont() object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or go_scattermap_marker_colorbar_tickformatstop() objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelstep Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_scattermap_marker_colorbar_title()` object.

`x` Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`xanchor` Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`y` Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

`yanchor` Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

side — Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.

Details

`go_scattermap()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_scattermap()` creates an object and adds it to a figure.

Nested objects documented here:

`go_scattermap_cluster()` `scattermap.cluster`

`go_scattermap_hoverlabel()` `scattermap.hoverlabel`

`go_scattermap_hoverlabel_font()` `scattermap.hoverlabel.font`: Sets the font used in hover labels.

`go_scattermap_legendgrouptitle()` `scattermap.legendgrouptitle`

`go_scattermap_legendgrouptitle_font()` `scattermap.legendgrouptitle.font`: Sets this legend group's title font.

`go_scattermap_line()` `scattermap.line`

`go_scattermap_marker()` `scattermap.marker`

`go_scattermap_marker_colorbar()` `scattermap.marker.colorbar`

`go_scattermap_marker_colorbar_tickfont()` `scattermap.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_scattermap_marker_colorbar_tickformatstop()` `scattermap.marker.colorbar.tickformatstop`

`go_scattermap_marker_colorbar_title()` `scattermap.marker.colorbar.title`

`go_scattermap_marker_colorbar_title_font()` `scattermap.marker.colorbar.title.font`: Sets this color bar's title font.

`go_scattermap_selected()` `scattermap.selected`

`go_scattermap_selected_marker()` `scattermap.selected.marker`

`go_scattermap_textfont()` `scattermap.textfont`: Sets the icon text font (color=map.layer.paint.text-color, size=map.layer.layout.text-size). Has an effect only when ‘type‘ is set to "symbol".

`go_scattermap_unselected()` `scattermap.unselected`

`go_scattermap_unselected_marker()` `scattermap.unselected.marker`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_scattermap(mode = "markers", marker_color = "red")
```

`add_scatterpolar` *Scatterpolar traces*

Description

The scatterpolar trace type encompasses line charts, scatter charts, text charts, and bubble charts in polar coordinates. The data visualized as scatter point or lines is set in ‘r‘ (radial) and ‘theta‘ (angular) coordinates Text (appearing either on the chart or on hover only) is via ‘text‘. Bubble charts are achieved by setting ‘marker.size‘ and/or ‘marker.color‘ to numerical arrays.

Usage

```
add_scatterpolar(fig, ...)

## Default S3 method:
add_scatterpolar(
  fig,
  ...,
  data = NULL,
  cliponaxis = NULL,
  connectgaps = NULL,
  customdata = NULL,
  dr = NULL,
  dtheta = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
```

```
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  r = NULL,
  r0 = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  subplot = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  theta = NULL,
  theta0 = NULL,
  thetaunit = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_scatterpolar(
  arg = NULL,
  ...,
  cliponaxis = NULL,
  connectgaps = NULL,
  customdata = NULL,
  dr = NULL,
  dtheta = NULL,
  fill = NULL,
```

```
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  r = NULL,
  r0 = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  subplot = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  theta = NULL,
  theta0 = NULL,
  thetaunit = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
```

```
  showarrow = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_line(
  arg = NULL,
  ...,
  backoff = NULL,
  color = NULL,
  dash = NULL,
```

```
  shape = NULL,
  smoothing = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_marker(
  arg = NULL,
  ...,
  angle = NULL,
  angleref = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  gradient = NULL,
  line = NULL,
  maxdisplayed = NULL,
  opacity = NULL,
  reversescale = NULL,
  showscale = NULL,
  size = NULL,
  sizemin = NULL,
  sizemode = NULL,
  sizeref = NULL,
  standoff = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
```

```
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
```

```
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_marker_gradient(
  arg = NULL,
  ...,
  color = NULL,
  type = NULL,
  skip_invalid = FALSE
)
```

```
go_scatterpolar_marker_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  dash = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_selected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_selected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_selected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
```

```
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scatterpolar_unselected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
cliponaxis	Determines whether or not markers and text nodes are clipped about the subplot axes. To show markers and text nodes above axis lines and tick labels, make sure to set ‘xaxis.layer‘ and ‘yaxis.layer‘ to "below traces". Logical.
connectgaps	Determines whether or not gaps (i.e. {nan} or missing values) in the provided data arrays are connected. Logical.

customdata — Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

dr — Sets the r coordinate step. Number.

dtheta — Sets the theta coordinate step. By default, the ‘dtheta‘ step equals the subplot’s period divided by the length of the ‘r‘ coordinates. Number.

fill — Sets the area to fill with a solid color. Use with ‘fillcolor‘ if not "none". scatterpolar has a subset of the options available to scatter. "toself" connects the endpoints of the trace (or each segment of the trace if it has gaps) into a closed shape. "tonext" fills the space between two traces if one completely encloses the other (eg consecutive contour lines), and behaves like "toself" if there is no trace before it. "tonext" should not be used if one trace does not enclose the other. One of `"none"`, `"toself"`, `"tonext"`.

fillcolor — Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name.

hoverinfo — Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of `"r"`, `"theta"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

hoverlabel — Named list or `go_scatterpolar_hoverlabel()` object.

hoveron — Do the hover effects highlight individual points (markers or line points) or do they highlight filled regions? If the fill is "toself" or "tonext" and there are no markers or text, then the default is "fills", otherwise it is "points". Any combination of `"points"`, `"fills"` joined with `"+"`.

hovertemplate — Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format’s syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format’s syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can’t be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example ‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_scatterpolar_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line`
: `go_scatterpolar()`: Named list or `go_scatterpolar_line()` object.

  `go_scatterpolar_marker()`: Named list or `go_scatterpolar_marker_line()` object.

`marker`
: `go_scatterpolar()`: Named list or `go_scatterpolar_marker()` object.

  `go_scatterpolar_selected()`: Named list or `go_scatterpolar_selected_marker()` object.

  `go_scatterpolar_unselected()`: Named list or `go_scatterpolar_unselected_marker()` object.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

mode	Determines the drawing mode for this scatter trace. If the provided ‘mode‘ includes "text" then the ‘text‘ elements appear at the coordinates. Otherwise, the ‘text‘ elements appear on hover. If there are less than 20 points and the trace is not stacked then the default is "lines+markers". Otherwise, "lines". Any combination of "lines", "markers", "text" joined with "+", or one of "none".
name	go_scatterpolar(): Sets the trace name. The trace name appears as the legend item and on hover. String. go_scatterpolar_marker_colorbar_tickformatstop(): When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.
opacity	go_scatterpolar(): Sets the opacity of the trace. Number in the interval [0, 1]. go_scatterpolar_marker(): Sets the marker opacity. Number in the interval [0, 1], or a vector of these. go_scatterpolar_selected_marker(): Sets the marker opacity of selected points. Number in the interval [0, 1]. go_scatterpolar_unselected_marker(): Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].
r	Sets the radial coordinates Data array: a vector, factor, Date/POSIXct vector, matrix or list.
r0	Alternate to ‘r‘. Builds a linear space of r coordinates. Use with ‘dr‘ where ‘r0‘ is the starting coordinate and ‘dr‘ the step. Any value.
selected	Named list or go_scatterpolar_selected() object.
selectedpoints	Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.
showlegend	Determines whether or not an item corresponding to this trace is shown in the legend. Logical.
subplot	Sets a reference between this trace’s data coordinates and a polar subplot. If "polar" (the default value), the data refer to ‘layout.polar‘. If "polar2", the data refer to ‘layout.polar2‘, and so on. Subplot id: "polar", "polar2", "polar3", ...
text	go_scatterpolar(): Sets text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace’s (x,y) coordinates. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these. go_scatterpolar_legendgrouptitle(): Sets the title of the legend group. String.

go_scatterpolar_marker_colorbar_title(): Sets the title of the color bar. String.

textfont
: go_scatterpolar(): Sets the text font. Named list or go_scatterpolar_textfont() object.
go_scatterpolar_selected(): Named list or go_scatterpolar_selected_textfont() object.
go_scatterpolar_unselected(): Named list or go_scatterpolar_unselected_textfont() object.

textposition
: Sets the positions of the 'text' elements with respects to the (x,y) coordinates. One of "top left", "top center", "top right", "middle left", "middle center", "middle right", "bottom left", "bottom center", "bottom right", or a vector of these.

texttemplate
: Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'r', 'theta' and 'text'. String, or a vector of these.

texttemplatefallback
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

theta
: Sets the angular coordinates Data array: a vector, factor, Date/POSIXct vector, matrix or list.

theta0
: Alternate to 'theta'. Builds a linear space of theta coordinates. Use with 'dtheta' where 'theta0' is the starting coordinate and 'dtheta' the step. Any value.

thetaunit
: Sets the unit of input "theta" values. Has an effect only when on "linear" angular axes. One of "radians", "degrees", "gradians".

uid
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision
: Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can

	add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.
unselected	Named list or go_scatterpolar_unselected() object.
visible	Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.
row	Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
align	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.
bgcolor	go_scatterpolar_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these. go_scatterpolar_marker_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name.
bordercolor	go_scatterpolar_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these. go_scatterpolar_marker_colorbar(): Sets the axis line color. Color: a CSS color string or R color name.
font	go_scatterpolar_hoverlabel(): Sets the font used in hover labels. Named list or go_scatterpolar_hoverlabel_font() object. go_scatterpolar_legendgrouptitle(): Sets this legend group's title font. Named list or go_scatterpolar_legendgrouptitle_font() object. go_scatterpolar_marker_colorbar_title(): Sets this color bar's title font. Named list or go_scatterpolar_marker_colorbar_title_font() object.
namelength	Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.
showarrow	Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.
color	go_scatterpolar_hoverlabel_font(), go_scatterpolar_textfont(): Color: a CSS color string or R color name, or a vector of these. go_scatterpolar_legendgrouptitle_font(), go_scatterpolar_marker_colorbar_tickfont(), go_scatterpolar_marker_colorbar_title_font(): Color: a CSS color string or R color name.

go_scatterpolar_line(): Sets the line color. Color: a CSS color string or R color name.

go_scatterpolar_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.cmin‘ and ‘marker.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scatterpolar.marker.colorscale, or a vector of these.

go_scatterpolar_marker_gradient(): Sets the final color of the gradient fill: the center color for radial, the right for horizontal, or the bottom for vertical. Color: a CSS color string or R color name, or a vector of these.

go_scatterpolar_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scatterpolar.marker.line.colorscale, or a vector of these.

go_scatterpolar_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name.

go_scatterpolar_selected_textfont(): Sets the text font color of selected points. Color: a CSS color string or R color name.

go_scatterpolar_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

go_scatterpolar_unselected_textfont(): Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

family

go_scatterpolar_hoverlabel_font(),go_scatterpolar_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

go_scatterpolar_legendgrouptitle_font(),go_scatterpolar_marker_colorbar_tickfont(), go_scatterpolar_marker_colorbar_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition

go_scatterpolar_hoverlabel_font(),go_scatterpolar_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_scatterpolar_legendgrouptitle_font(),go_scatterpolar_marker_colorbar_tickfont(), go_scatterpolar_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow
: go_scatterpolar_hoverlabel_font(),go_scatterpolar_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.
go_scatterpolar_legendgrouptitle_font(),go_scatterpolar_marker_colorbar_tickfont(), go_scatterpolar_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: go_scatterpolar_hoverlabel_font(),go_scatterpolar_textfont(): Number in the interval [1, Inf], or a vector of these.
go_scatterpolar_legendgrouptitle_font(),go_scatterpolar_marker_colorbar_tickfont(), go_scatterpolar_marker_colorbar_title_font(): Number in the interval [1, Inf].
go_scatterpolar_marker(): Sets the marker size (in px). Number in the interval [0, Inf], or a vector of these.
go_scatterpolar_selected_marker(): Sets the marker size of selected points. Number in the interval [0, Inf].
go_scatterpolar_unselected_marker(): Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

style
: go_scatterpolar_hoverlabel_font(),go_scatterpolar_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.
go_scatterpolar_legendgrouptitle_font(),go_scatterpolar_marker_colorbar_tickfont(), go_scatterpolar_marker_colorbar_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_scatterpolar_hoverlabel_font(),go_scatterpolar_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.
go_scatterpolar_legendgrouptitle_font(),go_scatterpolar_marker_colorbar_tickfont(), go_scatterpolar_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_scatterpolar_hoverlabel_font(),go_scatterpolar_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.
go_scatterpolar_legendgrouptitle_font(),go_scatterpolar_marker_colorbar_tickfont(), go_scatterpolar_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_scatterpolar_hoverlabel_font(),go_scatterpolar_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.
go_scatterpolar_legendgrouptitle_font(),go_scatterpolar_marker_colorbar_tickfont(), go_scatterpolar_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

backoff
: Sets the line back off from the end point of the nth line segment (in px). This option is useful e.g. to avoid overlap with arrowhead markers. With "auto" the lines would trim before markers if ‘marker.angleref‘ is set to "previous". Number in the interval [0, Inf], or a vector of these.

dash
: `go_scatterpolar_line()`: Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\`

  `go_scatterpolar_marker_line()`: Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\` or a vector of these.

shape
: Determines the line shape. With "spline" the lines are drawn using spline interpolation. The other available values correspond to step-wise line shapes. One of `"linear"`, `"spline"`.

smoothing
: Has an effect only if ‘shape‘ is set to "spline" Sets the amount of smoothing. 0 corresponds to no smoothing (equivalent to a "linear" shape). Number in the interval [0, 1.3].

width
: `go_scatterpolar_line()`: Sets the line width (in px). Number in the interval [0, Inf].

  `go_scatterpolar_marker_line()`: Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

angle
: Sets the marker angle in respect to ‘angleref‘. Angle in degrees, or a vector of these.

angleref
: Sets the reference for marker angle. With "previous", angle 0 points along the line from the previous point to this one. With "up", angle 0 points toward the top of the screen. One of `"previous"`, `"up"`.

autocolorscale
: `go_scatterpolar_marker()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

  `go_scatterpolar_marker_line()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

cauto
: `go_scatterpolar_marker()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

go_scatterpolar_marker_line(): Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

cmax

go_scatterpolar_marker(): Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

go_scatterpolar_marker_line(): Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

cmid

go_scatterpolar_marker(): Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

go_scatterpolar_marker_line(): Sets the mid-point of the color domain by scaling ‘marker.line.cmin‘ and/or ‘marker.line.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

cmin

go_scatterpolar_marker(): Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

go_scatterpolar_marker_line(): Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

coloraxis

Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: "coloraxis", "coloraxis2", "coloraxis3", ...

colorbar

Named list or go_scatterpolar_marker_colorbar() object.

colorscale

go_scatterpolar_marker(): Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rai
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the

name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

go_scatterpolar_marker_line(): Sets the colorscale. Has an effect only if in 'marker.line.color' is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the bounds of the colorscale in color space, use 'marker.line.cmin' and 'marker.line.cmax'. Alternatively, 'colorscale' may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,H
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

gradient — Named list or go_scatterpolar_marker_gradient() object.

maxdisplayed — Sets a maximum number of points to be drawn on the graph. 0 corresponds to no limit. Number in the interval [0, Inf].

reversescale — go_scatterpolar_marker(): Reverses the color mapping if true. Has an effect only if in 'marker.color' is set to a numerical array. If true, 'marker.cmin' will correspond to the last color in the array and 'marker.cmax' will correspond to the first color. Logical.

go_scatterpolar_marker_line(): Reverses the color mapping if true. Has an effect only if in 'marker.line.color' is set to a numerical array. If true, 'marker.line.cmin' will correspond to the last color in the array and 'marker.line.cmax' will correspond to the first color. Logical.

showscale — Determines whether or not a colorbar is displayed for this trace. Has an effect only if in 'marker.color' is set to a numerical array. Logical.

sizemin — Has an effect only if 'marker.size' is set to a numerical array. Sets the minimum size (in px) of the rendered marker points. Number in the interval [0, Inf].

sizemode — Has an effect only if 'marker.size' is set to a numerical array. Sets the rule for which the data in 'size' is converted to pixels. One of "diameter", "area".

sizeref — Has an effect only if 'marker.size' is set to a numerical array. Sets the scale factor used to determine the rendered size of marker points. Use with 'sizemin' and 'sizemode'. Number.

standoff — Moves the marker away from the data point in the direction of 'angle' (in px). This can be useful for example if you have another marker at this location and you want to point an arrowhead marker at it. Number in the interval [0, Inf], or a vector of these.

symbol — Sets the marker symbol type. Adding 100 is equivalent to appending "-open" to a symbol name. Adding 200 is equivalent to appending "-dot" to a symbol name. Adding 300 is equivalent to appending "-open-dot" or "dot-open" to a symbol name. One of "0", "circle", "100", "circle-open", "200", "circle-dot", "300", "circle-open-dot", "1", "square", "101", "square-open", "201", "square-dot", "301", "square-open-dot", "2", "diamond", "102", "diamond-open", "202", "diamond-dot", "302", "diamond-open-dot", "3", "cross", "103", "cross-open", "203", "cross-dot", "303", "cross-open-dot", "4", "x", "104", "x-open", "204", "x-dot", "304", "x-open-dot", "5", "triangle-up",

"105", "triangle-up-open", "205", "triangle-up-dot", "305", "triangle-up-open-dot",
"6", "triangle-down", "106", "triangle-down-open", "206", "triangle-down-dot",
"306", "triangle-down-open-dot", "7", "triangle-left", "107", "triangle-left-open",
"207", "triangle-left-dot", "307", "triangle-left-open-dot", "8", "triangle-right",
"108", "triangle-right-open", "208", "triangle-right-dot", "308", "triangle-right-open-do
"9", "triangle-ne", "109", "triangle-ne-open", "209", "triangle-ne-dot",
"309", "triangle-ne-open-dot", "10", "triangle-se", "110", "triangle-se-open",
"210", "triangle-se-dot", "310", "triangle-se-open-dot", "11", "triangle-sw",
"111", "triangle-sw-open", "211", "triangle-sw-dot", "311", "triangle-sw-open-dot",
"12", "triangle-nw", "112", "triangle-nw-open", "212", "triangle-nw-dot",
"312", "triangle-nw-open-dot", "13", "pentagon", "113", "pentagon-open",
"213", "pentagon-dot", "313", "pentagon-open-dot", "14", "hexagon",
"114", "hexagon-open", "214", "hexagon-dot", "314", "hexagon-open-dot",
"15", "hexagon2", "115", "hexagon2-open", "215", "hexagon2-dot", "315",
"hexagon2-open-dot", "16", "octagon", "116", "octagon-open", "216",
"octagon-dot", "316", "octagon-open-dot", "17", "star", "117", "star-open",
"217", "star-dot", "317", "star-open-dot", "18", "hexagram", "118", "hexagram-open",
"218", "hexagram-dot", "318", "hexagram-open-dot", "19", "star-triangle-up",
"119", "star-triangle-up-open", "219", "star-triangle-up-dot", "319",
"star-triangle-up-open-dot", "20", "star-triangle-down", "120", "star-triangle-down-open
"220", "star-triangle-down-dot", "320", "star-triangle-down-open-dot",
"21", "star-square", "121", "star-square-open", "221", "star-square-dot",
"321", "star-square-open-dot", "22", "star-diamond", "122", "star-diamond-open",
"222", "star-diamond-dot", "322", "star-diamond-open-dot", "23", "diamond-tall",
"123", "diamond-tall-open", "223", "diamond-tall-dot", "323", "diamond-tall-open-dot",
"24", "diamond-wide", "124", "diamond-wide-open", "224", "diamond-wide-dot",
"324", "diamond-wide-open-dot", "25", "hourglass", "125", "hourglass-open",
"26", "bowtie", "126", "bowtie-open", "27", "circle-cross", "127", "circle-cross-open",
"28", "circle-x", "128", "circle-x-open", "29", "square-cross", "129",
"square-cross-open", "30", "square-x", "130", "square-x-open", "31",
"diamond-cross", "131", "diamond-cross-open", "32", "diamond-x", "132",
"diamond-x-open", "33", "cross-thin", "133", "cross-thin-open", "34",
"x-thin", "134", "x-thin-open", "35", "asterisk", "135", "asterisk-open",
"36", "hash", "136", "hash-open", "236", "hash-dot", "336", "hash-open-dot",
"37", "y-up", "137", "y-up-open", "38", "y-down", "138", "y-down-open",
"39", "y-left", "139", "y-left-open", "40", "y-right", "140", "y-right-open",
"41", "line-ew", "141", "line-ew-open", "42", "line-ns", "142", "line-ns-open",
"43", "line-ne", "143", "line-ne-open", "44", "line-nw", "144", "line-nw-open",
"45", "arrow-up", "145", "arrow-up-open", "46", "arrow-down", "146",
"arrow-down-open", "47", "arrow-left", "147", "arrow-left-open", "48",
"arrow-right", "148", "arrow-right-open", "49", "arrow-bar-up", "149",
"arrow-bar-up-open", "50", "arrow-bar-down", "150", "arrow-bar-down-open",
"51", "arrow-bar-left", "151", "arrow-bar-left-open", "52", "arrow-bar-right",
"152", "arrow-bar-right-open", "53", "arrow", "153", "arrow-open", "54",
"arrow-wide", "154", "arrow-wide-open", 0, 100, 200, 300, 1, 101, 201, 301,
2, 102, 202, 302, 3, 103, 203, 303, 4, 104, 204, 304, 5, 105, 205, 305, 6, 106,
206, 306, 7, 107, 207, 307, 8, 108, 208, 308, 9, 109, 209, 309, 10, 110, 210, 310,
11, 111, 211, 311, 12, 112, 212, 312, 13, 113, 213, 313, 14, 114, 214, 314, 15,

115, 215, 315, 16, 116, 216, 316, 17, 117, 217, 317, 18, 118, 218, 318, 19, 119, 219, 319, 20, 120, 220, 320, 21, 121, 221, 321, 22, 122, 222, 322, 23, 123, 223, 323, 24, 124, 224, 324, 25, 125, 26, 126, 27, 127, 28, 128, 29, 129, 30, 130, 31, 131, 32, 132, 33, 133, 34, 134, 35, 135, 36, 136, 236, 336, 37, 137, 38, 138, 39, 139, 40, 140, 41, 141, 42, 142, 43, 143, 44, 144, 45, 145, 46, 146, 47, 147, 48, 148, 49, 149, 50, 150, 51, 151, 52, 152, 53, 153, 54, 154, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick

is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".

showticksuffix Same as 'showtickprefix' but for tick suffixes. One of "all", "first", "last", "none".

thickness Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of "fraction", "pixels".

tick0 Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar's tick label font Named list or go_scatterpolar_marker_colorbar_tickfont() object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or go_scatterpolar_marker_colorbar_tickformatstop() objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_scatterpolar_marker_colorbar_title()` object.

`x` Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`xanchor` Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`y` Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

yanchor — Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

side — Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of "right", "top", "bottom".

type — Sets the type of gradient used to fill the markers One of "radial", "horizontal", "vertical", "none", or a vector of these.

Details

`go_scatterpolar()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_scatterpolar()` creates an object and adds it to a figure.

Nested objects documented here:

`go_scatterpolar_hoverlabel()` `scatterpolar.hoverlabel`

`go_scatterpolar_hoverlabel_font()` `scatterpolar.hoverlabel.font`: Sets the font used in hover labels.

`go_scatterpolar_legendgrouptitle()` `scatterpolar.legendgrouptitle`

`go_scatterpolar_legendgrouptitle_font()` `scatterpolar.legendgrouptitle.font`: Sets this legend group's title font.

`go_scatterpolar_line()` `scatterpolar.line`

`go_scatterpolar_marker()` `scatterpolar.marker`

`go_scatterpolar_marker_colorbar()` `scatterpolar.marker.colorbar`

`go_scatterpolar_marker_colorbar_tickfont()` `scatterpolar.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_scatterpolar_marker_colorbar_tickformatstop()` `scatterpolar.marker.colorbar.tickformatstop`

`go_scatterpolar_marker_colorbar_title()` `scatterpolar.marker.colorbar.title`

`go_scatterpolar_marker_colorbar_title_font()` `scatterpolar.marker.colorbar.title.font`: Sets this color bar's title font.

`go_scatterpolar_marker_gradient()` `scatterpolar.marker.gradient`

`go_scatterpolar_marker_line()` `scatterpolar.marker.line`

`go_scatterpolar_selected()` `scatterpolar.selected`

`go_scatterpolar_selected_marker()` `scatterpolar.selected.marker`

`go_scatterpolar_selected_textfont()` `scatterpolar.selected.textfont`

`go_scatterpolar_textfont()` `scatterpolar.textfont`: Sets the text font.

`go_scatterpolar_unselected()` `scatterpolar.unselected`

`go_scatterpolar_unselected_marker()` `scatterpolar.unselected.marker`

`go_scatterpolar_unselected_textfont()` `scatterpolar.unselected.textfont`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_scatterpolar(mode = "markers", marker_color = "red")
```

---

add_scatterpolargl *Scatterpolargl traces*

Description

The scatterpolargl trace type encompasses line charts, scatter charts, and bubble charts in polar coordinates using the WebGL plotting engine. The data visualized as scatter point or lines is set in 'r' (radial) and 'theta' (angular) coordinates Bubble charts are achieved by setting 'marker.size' and/or 'marker.color' to numerical arrays.

Usage

```
add_scatterpolargl(fig, ...)

## Default S3 method:
add_scatterpolargl(
  fig,
  ...,
  data = NULL,
  connectgaps = NULL,
  customdata = NULL,
  dr = NULL,
  dtheta = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  r = NULL,
  r0 = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  subplot = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  theta = NULL,
  theta0 = NULL,
  thetaunit = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
```

```
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl(
  arg = NULL,
  ...,
  connectgaps = NULL,
  customdata = NULL,
  dr = NULL,
  dtheta = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  r = NULL,
  r0 = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  subplot = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  theta = NULL,
  theta0 = NULL,
  thetaunit = NULL,
  uid = NULL,
  uirevision = NULL,
```

```
  unselected = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
```

```
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_line(
  arg = NULL,
  ...,
  color = NULL,
  dash = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_marker(
  arg = NULL,
  ...,
  angle = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  line = NULL,
  opacity = NULL,
  reversescale = NULL,
  showscale = NULL,
  size = NULL,
  sizemin = NULL,
  sizemode = NULL,
  sizeref = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
```

```
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_marker_colorbar_tickfont(
  arg = NULL,
  ...,
```

```
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_marker_line(
```

```
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_selected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_selected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_selected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  size = NULL,
  style = NULL,
  variant = NULL,
  weight = NULL,
```

```
  skip_invalid = FALSE
)

go_scatterpolargl_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scatterpolargl_unselected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
connectgaps	Determines whether or not gaps (i.e. {nan} or missing values) in the provided data arrays are connected. Logical.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
dr	Sets the r coordinate step. Number.
dtheta	Sets the theta coordinate step. By default, the ‘dtheta‘ step equals the subplot's period divided by the length of the ‘r‘ coordinates. Number.

`fill` Sets the area to fill with a solid color. Defaults to "none" unless this trace is stacked, then it gets "tonexty" ("tonextx") if 'orientation' is "v" ("h") Use with 'fillcolor' if not "none". "tozerox" and "tozeroy" fill to x=0 and y=0 respectively. "tonextx" and "tonexty" fill between the endpoints of this trace and the endpoints of the trace before it, connecting those endpoints with straight lines (to make a stacked area graph); if there is no trace before it, they behave like "tozerox" and "tozeroy". "toself" connects the endpoints of the trace (or each segment of the trace if it has gaps) into a closed shape. "tonext" fills the space between two traces if one completely encloses the other (eg consecutive contour lines), and behaves like "toself" if there is no trace before it. "tonext" should not be used if one trace does not enclose the other. Traces in a 'stackgroup' will only fill to (or be filled to) other traces in the same group. With multiple 'stackgroup's or some traces stacked and some not, if fill-linked traces are not already consecutive, the later ones will be pushed down in the drawing order. One of `"none"`, `"tozeroy"`, `"tozerox"`, `"tonexty"`, `"tonextx"`, `"toself"`, `"tonext"`.

`fillcolor` Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name.

`hoverinfo` Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of `"r"`, `"theta"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_scatterpolargl_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Sets hover text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
Named list or `go_scatterpolargl_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line` `go_scatterpolargl()`: Named list or `go_scatterpolargl_line()` object.
`go_scatterpolargl_marker()`: Named list or `go_scatterpolargl_marker_line()` object.

`marker` `go_scatterpolargl()`: Named list or `go_scatterpolargl_marker()` object.
`go_scatterpolargl_selected()`: Named list or `go_scatterpolargl_selected_marker()` object.
`go_scatterpolargl_unselected()`: Named list or `go_scatterpolargl_unselected_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

`mode` Determines the drawing mode for this scatter trace. If the provided 'mode' includes "text" then the 'text' elements appear at the coordinates. Otherwise, the 'text' elements appear on hover. If there are less than 20 points and the trace is not stacked then the default is "lines+markers". Otherwise, "lines". Any combination of `"lines"`, `"markers"`, `"text"` joined with `"+"`, or one of `"none"`.

name	go_scatterpolargl(): Sets the trace name. The trace name appears as the legend item and on hover. String. go_scatterpolargl_marker_colorbar_tickformatstop(): When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.
opacity	go_scatterpolargl(): Sets the opacity of the trace. Number in the interval [0, 1]. go_scatterpolargl_marker(): Sets the marker opacity. Number in the interval [0, 1], or a vector of these. go_scatterpolargl_selected_marker(): Sets the marker opacity of selected points. Number in the interval [0, 1]. go_scatterpolargl_unselected_marker(): Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].
r	Sets the radial coordinates Data array: a vector, factor, Date/POSIXct vector, matrix or list.
r0	Alternate to 'r'. Builds a linear space of r coordinates. Use with 'dr' where 'r0' is the starting coordinate and 'dr' the step. Any value.
selected	Named list or go_scatterpolargl_selected() object.
selectedpoints	Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the 'unselected' are turned on for all points, whereas, any other non-array values means no selection all where the 'selected' and 'unselected' styles have no effect. Any value.
showlegend	Determines whether or not an item corresponding to this trace is shown in the legend. Logical.
subplot	Sets a reference between this trace's data coordinates and a polar subplot. If "polar" (the default value), the data refer to 'layout.polar'. If "polar2", the data refer to 'layout.polar2', and so on. Subplot id: "polar", "polar2", "polar3", ...
text	go_scatterpolargl(): Sets text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. If trace 'hoverinfo' contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these. go_scatterpolargl_legendgrouptitle(): Sets the title of the legend group. String. go_scatterpolargl_marker_colorbar_title(): Sets the title of the color bar. String.
textfont	go_scatterpolargl(): Sets the text font. Named list or go_scatterpolargl_textfont() object.

go_scatterpolargl_selected(): Named list or go_scatterpolargl_selected_textfont() object.

go_scatterpolargl_unselected(): Named list or go_scatterpolargl_unselected_textfont() object.

textposition — Sets the positions of the 'text' elements with respects to the (x,y) coordinates. One of "top left", "top center", "top right", "middle left", "middle center", "middle right", "bottom left", "bottom center", "bottom right", or a vector of these.

texttemplate — Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'r', 'theta' and 'text'. String, or a vector of these.

texttemplatefallback — Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

theta — Sets the angular coordinates Data array: a vector, factor, Date/POSIXct vector, matrix or list.

theta0 — Alternate to 'theta'. Builds a linear space of theta coordinates. Use with 'dtheta' where 'theta0' is the starting coordinate and 'dtheta' the step. Any value.

thetaunit — Sets the unit of input "theta" values. Has an effect only when on "linear" angular axes. One of "radians", "degrees", "gradians".

uid — Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision — Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

unselected — Named list or go_scatterpolargl_unselected() object.

visible — Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.

row — Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.

col — Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

skip_invalid — If TRUE, silently drop invalid properties instead of raising an error.

arg — Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

align — Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.

bgcolor — go_scatterpolargl_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

go_scatterpolargl_marker_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name.

bordercolor — go_scatterpolargl_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

go_scatterpolargl_marker_colorbar(): Sets the axis line color. Color: a CSS color string or R color name.

font — go_scatterpolargl_hoverlabel(): Sets the font used in hover labels. Named list or go_scatterpolargl_hoverlabel_font() object.

go_scatterpolargl_legendgrouptitle(): Sets this legend group's title font. Named list or go_scatterpolargl_legendgrouptitle_font() object.

go_scatterpolargl_marker_colorbar_title(): Sets this color bar's title font. Named list or go_scatterpolargl_marker_colorbar_title_font() object.

namelength — Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow — Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color — go_scatterpolargl_hoverlabel_font(), go_scatterpolargl_textfont(): Color: a CSS color string or R color name, or a vector of these.

go_scatterpolargl_legendgrouptitle_font(), go_scatterpolargl_marker_colorbar_tickfont
go_scatterpolargl_marker_colorbar_title_font(): Color: a CSS color string or R color name.

go_scatterpolargl_line(): Sets the line color. Color: a CSS color string or R color name.

go_scatterpolargl_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.cmin‘ and ‘marker.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scatterpolargl.marker.colorscale, or a vector of these.

go_scatterpolargl_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scatterpolargl.marker.line.colorscale, or a vector of these.

go_scatterpolargl_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name.

go_scatterpolargl_selected_textfont(): Sets the text font color of selected points. Color: a CSS color string or R color name.

go_scatterpolargl_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

go_scatterpolargl_unselected_textfont(): Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

family

go_scatterpolargl_hoverlabel_font(),go_scatterpolargl_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

go_scatterpolargl_legendgrouptitle_font(),go_scatterpolargl_marker_colorbar_tickfont
go_scatterpolargl_marker_colorbar_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition

go_scatterpolargl_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_scatterpolargl_legendgrouptitle_font(),go_scatterpolargl_marker_colorbar_tickfont
go_scatterpolargl_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow

go_scatterpolargl_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_scatterpolargl_legendgrouptitle_font(),go_scatterpolargl_marker_colorbar_tickfont
go_scatterpolargl_marker_colorbar_title_font(): Sets the shape and

color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

`size` `go_scatterpolargl_hoverlabel_font()`, `go_scatterpolargl_textfont()`: Number in the interval [1, Inf], or a vector of these.

`go_scatterpolargl_legendgrouptitle_font()`, `go_scatterpolargl_marker_colorbar_tickfont` `go_scatterpolargl_marker_colorbar_title_font()`: Number in the interval [1, Inf].

`go_scatterpolargl_marker()`: Sets the marker size (in px). Number in the interval [0, Inf], or a vector of these.

`go_scatterpolargl_selected_marker()`: Sets the marker size of selected points. Number in the interval [0, Inf].

`go_scatterpolargl_unselected_marker()`: Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

`style` `go_scatterpolargl_hoverlabel_font()`, `go_scatterpolargl_textfont()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`, or a vector of these.

`go_scatterpolargl_legendgrouptitle_font()`, `go_scatterpolargl_marker_colorbar_tickfont` `go_scatterpolargl_marker_colorbar_title_font()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

`textcase` `go_scatterpolargl_hoverlabel_font()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`, or a vector of these.

`go_scatterpolargl_legendgrouptitle_font()`, `go_scatterpolargl_marker_colorbar_tickfont` `go_scatterpolargl_marker_colorbar_title_font()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.

`variant` `go_scatterpolargl_hoverlabel_font()`: Sets the variant of the font. One of `"normal"`, `"small-caps"`, `"all-small-caps"`, `"all-petite-caps"`, `"petite-caps"`, `"unicase"`, or a vector of these.

`go_scatterpolargl_legendgrouptitle_font()`, `go_scatterpolargl_marker_colorbar_tickfont` `go_scatterpolargl_marker_colorbar_title_font()`: Sets the variant of the font. One of `"normal"`, `"small-caps"`, `"all-small-caps"`, `"all-petite-caps"`, `"petite-caps"`, `"unicase"`.

`go_scatterpolargl_textfont()`: Sets the variant of the font. One of `"normal"`, `"small-caps"`, or a vector of these.

`weight` `go_scatterpolargl_hoverlabel_font()`: Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

`go_scatterpolargl_legendgrouptitle_font()`, `go_scatterpolargl_marker_colorbar_tickfont` `go_scatterpolargl_marker_colorbar_title_font()`: Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

`go_scatterpolargl_textfont()`: Sets the weight (or boldness) of the font. One of `"normal"`, `"bold"`, or a vector of these.

dash — Sets the style of the lines. One of "dash", "dashdot", "dot", "longdash", "longdashdot", "solid".

width — go_scatterpolargl_line(): Sets the line width (in px). Number in the interval [0, Inf].

go_scatterpolargl_marker_line(): Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

angle — Sets the marker angle in respect to 'angleref'. Angle in degrees, or a vector of these.

autocolorscale — go_scatterpolargl_marker(): Determines whether the colorscale is a default palette ('autocolorscale: true') or the palette determined by 'marker.colorscale'. Has an effect only if in 'marker.color' is set to a numerical array. In case 'colorscale' is unspecified or 'autocolorscale' is true, the default palette will be chosen according to whether numbers in the 'color' array are all positive, all negative or mixed. Logical.

go_scatterpolargl_marker_line(): Determines whether the colorscale is a default palette ('autocolorscale: true') or the palette determined by 'marker.line.colorscale'. Has an effect only if in 'marker.line.color' is set to a numerical array. In case 'colorscale' is unspecified or 'autocolorscale' is true, the default palette will be chosen according to whether numbers in the 'color' array are all positive, all negative or mixed. Logical.

cauto — go_scatterpolargl_marker(): Determines whether or not the color domain is computed with respect to the input data (here in 'marker.color') or the bounds set in 'marker.cmin' and 'marker.cmax' Has an effect only if in 'marker.color' is set to a numerical array. Defaults to 'false' when 'marker.cmin' and 'marker.cmax' are set by the user. Logical.

go_scatterpolargl_marker_line(): Determines whether or not the color domain is computed with respect to the input data (here in 'marker.line.color') or the bounds set in 'marker.line.cmin' and 'marker.line.cmax' Has an effect only if in 'marker.line.color' is set to a numerical array. Defaults to 'false' when 'marker.line.cmin' and 'marker.line.cmax' are set by the user. Logical.

cmax — go_scatterpolargl_marker(): Sets the upper bound of the color domain. Has an effect only if in 'marker.color' is set to a numerical array. Value should have the same units as in 'marker.color' and if set, 'marker.cmin' must be set as well. Number.

go_scatterpolargl_marker_line(): Sets the upper bound of the color domain. Has an effect only if in 'marker.line.color' is set to a numerical array. Value should have the same units as in 'marker.line.color' and if set, 'marker.line.cmin' must be set as well. Number.

cmid — go_scatterpolargl_marker(): Sets the mid-point of the color domain by scaling 'marker.cmin' and/or 'marker.cmax' to be equidistant to this point. Has an effect only if in 'marker.color' is set to a numerical array. Value should have the same units as in 'marker.color'. Has no effect when 'marker.cauto' is 'false'. Number.

go_scatterpolargl_marker_line(): Sets the mid-point of the color domain by scaling 'marker.line.cmin' and/or 'marker.line.cmax' to be equidistant to this point. Has an effect only if in 'marker.line.color' is set to a numerical array.

Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

`cmin` `go_scatterpolargl_marker()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`go_scatterpolargl_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_scatterpolargl_marker_colorbar()` object.

`colorscale` `go_scatterpolargl_marker()`: Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rai
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_scatterpolargl_marker_line()`: Sets the colorscale. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.line.cmin‘ and ‘marker.line.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,H
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`reversescale` `go_scatterpolargl_marker()`: Reverses the color mapping if true. Has an effect only if in ‘marker.color‘ is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.

`go_scatterpolargl_marker_line()`: Reverses the color mapping if true. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. If true, ‘marker.line.cmin‘ will correspond to the last color in the array and ‘marker.line.cmax‘ will correspond to the first color. Logical.

showscale Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘marker.color‘ is set to a numerical array. Logical.

sizemin Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the minimum size (in px) of the rendered marker points. Number in the interval [0, Inf].

sizemode Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the rule for which the data in ‘size‘ is converted to pixels. One of "diameter", "area".

sizeref Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the scale factor used to determine the rendered size of marker points. Use with ‘sizemin‘ and ‘sizemode‘. Number.

symbol Sets the marker symbol type. Adding 100 is equivalent to appending "-open" to a symbol name. Adding 200 is equivalent to appending "-dot" to a symbol name. Adding 300 is equivalent to appending "-open-dot" or "dot-open" to a symbol name. One of "0", "circle", "100", "circle-open", "200", "circle-dot", "300", "circle-open-dot", "1", "square", "101", "square-open", "201", "square-dot", "301", "square-open-dot", "2", "diamond", "102", "diamond-open", "202", "diamond-dot", "302", "diamond-open-dot", "3", "cross", "103", "cross-open", "203", "cross-dot", "303", "cross-open-dot", "4", "x", "104", "x-open", "204", "x-dot", "304", "x-open-dot", "5", "triangle-up", "105", "triangle-up-open", "205", "triangle-up-dot", "305", "triangle-up-open-dot", "6", "triangle-down", "106", "triangle-down-open", "206", "triangle-down-dot", "306", "triangle-down-open-dot", "7", "triangle-left", "107", "triangle-left-open", "207", "triangle-left-dot", "307", "triangle-left-open-dot", "8", "triangle-right", "108", "triangle-right-open", "208", "triangle-right-dot", "308", "triangle-right-open-do "9", "triangle-ne", "109", "triangle-ne-open", "209", "triangle-ne-dot", "309", "triangle-ne-open-dot", "10", "triangle-se", "110", "triangle-se-open", "210", "triangle-se-dot", "310", "triangle-se-open-dot", "11", "triangle-sw", "111", "triangle-sw-open", "211", "triangle-sw-dot", "311", "triangle-sw-open-dot", "12", "triangle-nw", "112", "triangle-nw-open", "212", "triangle-nw-dot", "312", "triangle-nw-open-dot", "13", "pentagon", "113", "pentagon-open", "213", "pentagon-dot", "313", "pentagon-open-dot", "14", "hexagon", "114", "hexagon-open", "214", "hexagon-dot", "314", "hexagon-open-dot", "15", "hexagon2", "115", "hexagon2-open", "215", "hexagon2-dot", "315", "hexagon2-open-dot", "16", "octagon", "116", "octagon-open", "216", "octagon-dot", "316", "octagon-open-dot", "17", "star", "117", "star-open", "217", "star-dot", "317", "star-open-dot", "18", "hexagram", "118", "hexagram-open", "218", "hexagram-dot", "318", "hexagram-open-dot", "19", "star-triangle-up", "119", "star-triangle-up-open", "219", "star-triangle-up-dot", "319", "star-triangle-up-open-dot", "20", "star-triangle-down", "120", "star-triangle-down-open "220", "star-triangle-down-dot", "320", "star-triangle-down-open-dot", "21", "star-square", "121", "star-square-open", "221", "star-square-dot", "321", "star-square-open-dot", "22", "star-diamond", "122", "star-diamond-open", "222", "star-diamond-dot", "322", "star-diamond-open-dot", "23", "diamond-tall", "123", "diamond-tall-open", "223", "diamond-tall-dot", "323", "diamond-tall-open-dot", "24", "diamond-wide", "124", "diamond-wide-open", "224", "diamond-wide-dot", "324", "diamond-wide-open-dot", "25", "hourglass", "125", "hourglass-open", "26", "bowtie", "126", "bowtie-open", "27", "circle-cross", "127", "circle-cross-open", "28", "circle-x", "128", "circle-x-open", "29", "square-cross", "129", "square-cross-open", "30", "square-x", "130", "square-x-open", "31",

"diamond-cross", "131", "diamond-cross-open", "32", "diamond-x", "132", "diamond-x-open", "33", "cross-thin", "133", "cross-thin-open", "34", "x-thin", "134", "x-thin-open", "35", "asterisk", "135", "asterisk-open", "36", "hash", "136", "hash-open", "236", "hash-dot", "336", "hash-open-dot", "37", "y-up", "137", "y-up-open", "38", "y-down", "138", "y-down-open", "39", "y-left", "139", "y-left-open", "40", "y-right", "140", "y-right-open", "41", "line-ew", "141", "line-ew-open", "42", "line-ns", "142", "line-ns-open", "43", "line-ne", "143", "line-ne-open", "44", "line-nw", "144", "line-nw-open", "45", "arrow-up", "145", "arrow-up-open", "46", "arrow-down", "146", "arrow-down-open", "47", "arrow-left", "147", "arrow-left-open", "48", "arrow-right", "148", "arrow-right-open", "49", "arrow-bar-up", "149", "arrow-bar-up-open", "50", "arrow-bar-down", "150", "arrow-bar-down-open", "51", "arrow-bar-left", "151", "arrow-bar-left-open", "52", "arrow-bar-right", "152", "arrow-bar-right-open", "53", "arrow", "153", "arrow-open", "54", "arrow-wide", "154", "arrow-wide-open", 0, 100, 200, 300, 1, 101, 201, 301, 2, 102, 202, 302, 3, 103, 203, 303, 4, 104, 204, 304, 5, 105, 205, 305, 6, 106, 206, 306, 7, 107, 207, 307, 8, 108, 208, 308, 9, 109, 209, 309, 10, 110, 210, 310, 11, 111, 211, 311, 12, 112, 212, 312, 13, 113, 213, 313, 14, 114, 214, 314, 15, 115, 215, 315, 16, 116, 216, 316, 17, 117, 217, 317, 18, 118, 218, 318, 19, 119, 219, 319, 20, 120, 220, 320, 21, 121, 221, 321, 22, 122, 222, 322, 23, 123, 223, 323, 24, 124, 224, 324, 25, 125, 26, 126, 27, 127, 28, 128, 29, 129, 30, 130, 31, 131, 32, 132, 33, 133, 34, 134, 35, 135, 36, 136, 236, 336, 37, 137, 38, 138, 39, 139, 40, 140, 41, 141, 42, 142, 43, 143, 44, 144, 45, 145, 46, 146, 47, 147, 48, 148, 49, 149, 50, 150, 51, 151, 52, 152, 53, 153, 54, 154, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with 'tick0'. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

`labelalias` Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_scatterpolargl_marker_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter:

"%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops`
: List of named lists or `go_scatterpolargl_marker_colorbar_tickformatstop()` objects.

`ticklabeloverflow`
: Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition`
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep`
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

`ticklen`
: Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode`
: Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix`
: Sets a tick label prefix. String.

`ticks`
: Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix`
: Sets a tick label suffix. String.

`ticktext`
: Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals`
: Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth`
: Sets the tick width (in px). Number in the interval [0, Inf].

`title`
: String (the title text), named list or `go_scatterpolargl_marker_colorbar_title()` object.

x — Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

xanchor — Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

xpad — Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

y — Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

yanchor — Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

side — Determines the location of color bar’s title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.

Details

go_scatterpolargl() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_scatterpolargl() creates an object and adds it to a figure.

Nested objects documented here:

go_scatterpolargl_hoverlabel() scatterpolargl.hoverlabel

go_scatterpolargl_hoverlabel_font() scatterpolargl.hoverlabel.font: Sets the font used in hover labels.

go_scatterpolargl_legendgrouptitle() scatterpolargl.legendgrouptitle

go_scatterpolargl_legendgrouptitle_font() scatterpolargl.legendgrouptitle.font: Sets this legend group's title font.

go_scatterpolargl_line() scatterpolargl.line

go_scatterpolargl_marker() scatterpolargl.marker

go_scatterpolargl_marker_colorbar() scatterpolargl.marker.colorbar

go_scatterpolargl_marker_colorbar_tickfont() scatterpolargl.marker.colorbar.tickfont: Sets the color bar's tick label font

go_scatterpolargl_marker_colorbar_tickformatstop() scatterpolargl.marker.colorbar.tickformatstop

go_scatterpolargl_marker_colorbar_title() scatterpolargl.marker.colorbar.title

go_scatterpolargl_marker_colorbar_title_font() scatterpolargl.marker.colorbar.title.font: Sets this color bar's title font.

go_scatterpolargl_marker_line() scatterpolargl.marker.line

go_scatterpolargl_selected() scatterpolargl.selected

go_scatterpolargl_selected_marker() scatterpolargl.selected.marker

go_scatterpolargl_selected_textfont() scatterpolargl.selected.textfont

go_scatterpolargl_textfont() scatterpolargl.textfont: Sets the text font.

go_scatterpolargl_unselected() scatterpolargl.unselected

go_scatterpolargl_unselected_marker() scatterpolargl.unselected.marker

go_scatterpolargl_unselected_textfont() scatterpolargl.unselected.textfont

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_scatterpolargl(mode = "markers", marker_color = "red")
```

`add_scattersmith` *Scattersmith traces*

Description

The scattersmith trace type encompasses line charts, scatter charts, text charts, and bubble charts in smith coordinates. The data visualized as scatter point or lines is set in ‘real‘ and ‘imag‘ (imaginary) coordinates Text (appearing either on the chart or on hover only) is via ‘text‘. Bubble charts are achieved by setting ‘marker.size‘ and/or ‘marker.color‘ to numerical arrays.

Usage

```
add_scattersmith(fig, ...)

## Default S3 method:
add_scattersmith(
  fig,
  ...,
  data = NULL,
  cliponaxis = NULL,
  connectgaps = NULL,
  customdata = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  imag = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  real = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
```

```
  subplot = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_scattersmith(
  arg = NULL,
  ...,
  cliponaxis = NULL,
  connectgaps = NULL,
  customdata = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  imag = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  real = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  subplot = NULL,
  text = NULL,
```

```
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_scattersmith_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_scattersmith_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattersmith_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scattersmith_legendgrouptitle_font(
  arg = NULL,
```

```
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattersmith_line(
  arg = NULL,
  ...,
  backoff = NULL,
  color = NULL,
  dash = NULL,
  shape = NULL,
  smoothing = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scattersmith_marker(
  arg = NULL,
  ...,
  angle = NULL,
  angleref = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  gradient = NULL,
  line = NULL,
  maxdisplayed = NULL,
  opacity = NULL,
  reversescale = NULL,
  showscale = NULL,
  size = NULL,
  sizemin = NULL,
  sizemode = NULL,
```

```
  sizeref = NULL,
  standoff = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_scattersmith_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
```

```
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_scattersmith_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattersmith_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_scattersmith_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scattersmith_marker_colorbar_title_font(
  arg = NULL,
  ...,
```

```
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattersmith_marker_gradient(
  arg = NULL,
  ...,
  color = NULL,
  type = NULL,
  skip_invalid = FALSE
)

go_scattersmith_marker_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  dash = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scattersmith_selected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scattersmith_selected_marker(
  arg = NULL,
  ...,
```

```
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scattersmith_selected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

go_scattersmith_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scattersmith_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scattersmith_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scattersmith_unselected_textfont(
  arg = NULL,
  ...,
```

```
  color = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
cliponaxis	Determines whether or not markers and text nodes are clipped about the subplot axes. To show markers and text nodes above axis lines and tick labels, make sure to set ‘xaxis.layer‘ and ‘yaxis.layer‘ to "below traces". Logical.
connectgaps	Determines whether or not gaps (i.e. {nan} or missing values) in the provided data arrays are connected. Logical.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
fill	Sets the area to fill with a solid color. Use with ‘fillcolor‘ if not "none". scattersmith has a subset of the options available to scatter. "toself" connects the endpoints of the trace (or each segment of the trace if it has gaps) into a closed shape. "tonext" fills the space between two traces if one completely encloses the other (eg consecutive contour lines), and behaves like "toself" if there is no trace before it. "tonext" should not be used if one trace does not enclose the other. One of "none", "toself", "tonext".
fillcolor	Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name.
hoverinfo	Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of "real", "imag", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_scattersmith_hoverlabel() object.
hoveron	Do the hover effects highlight individual points (markers or line points) or do they highlight filled regions? If the fill is "toself" or "tonext" and there are no markers or text, then the default is "fills", otherwise it is "points". Any combination of "points", "fills" joined with "+".
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_},

{%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`imag`
: Sets the imaginary component of the data, in units of normalized impedance such that real=1, imag=0 is the center of the chart. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_scattersmith_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line` `go_scattersmith()`: Named list or `go_scattersmith_line()` object.

`go_scattersmith_marker()`: Named list or `go_scattersmith_marker_line()` object.

`marker` `go_scattersmith()`: Named list or `go_scattersmith_marker()` object.

`go_scattersmith_selected()`: Named list or `go_scattersmith_selected_marker()` object.

`go_scattersmith_unselected()`: Named list or `go_scattersmith_unselected_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`mode` Determines the drawing mode for this scatter trace. If the provided ‘mode‘ includes "text" then the ‘text‘ elements appear at the coordinates. Otherwise, the ‘text‘ elements appear on hover. If there are less than 20 points and the trace is not stacked then the default is "lines+markers". Otherwise, "lines". Any combination of `"lines"`, `"markers"`, `"text"` joined with `"+"`, or one of `"none"`.

`name` `go_scattersmith()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_scattersmith_marker_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` `go_scattersmith()`: Sets the opacity of the trace. Number in the interval [0, 1].

`go_scattersmith_marker()`: Sets the marker opacity. Number in the interval [0, 1], or a vector of these.

`go_scattersmith_selected_marker()`: Sets the marker opacity of selected points. Number in the interval [0, 1].

`go_scattersmith_unselected_marker()`: Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

`real` Sets the real component of the data, in units of normalized impedance such that real=1, imag=0 is the center of the chart. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`selected` Named list or `go_scattersmith_selected()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the 'unselected' are turned on for all points, whereas, any other non-array values means no selection all where the 'selected' and 'unselected' styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`subplot` Sets a reference between this trace's data coordinates and a smith subplot. If "smith" (the default value), the data refer to 'layout.smith'. If "smith2", the data refer to 'layout.smith2', and so on. Subplot id: `"smith"`, `"smith2"`, `"smith3"`, ...

`text` `go_scattersmith()`: Sets text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. If trace 'hoverinfo' contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_scattersmith_legendgrouptitle()`: Sets the title of the legend group. String.

`go_scattersmith_marker_colorbar_title()`: Sets the title of the color bar. String.

`textfont` `go_scattersmith()`: Sets the text font. Named list or `go_scattersmith_textfont()` object.

`go_scattersmith_selected()`: Named list or `go_scattersmith_selected_textfont()` object.

`go_scattersmith_unselected()`: Named list or `go_scattersmith_unselected_textfont()` object.

`textposition` Sets the positions of the 'text' elements with respects to the (x,y) coordinates. One of `"top left"`, `"top center"`, `"top right"`, `"middle left"`, `"middle center"`, `"middle right"`, `"bottom left"`, `"bottom center"`, `"bottom right"`, or a vector of these.

`texttemplate` Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'real', 'imag' and 'text'. String, or a vector of these.

`texttemplatefallback` Fallback string that's displayed when a variable referenced in a template is miss-

ing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(x|y)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`unselected` Named list or `go_scattersmith_unselected()` object.

`visible` Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`row` Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`col` Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`skip_invalid` If `TRUE`, silently drop invalid properties instead of raising an error.

`arg` Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

`align` Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`bgcolor` `go_scattersmith_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`go_scattersmith_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

`bordercolor` `go_scattersmith_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`go_scattersmith_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

`font` `go_scattersmith_hoverlabel()`: Sets the font used in hover labels. Named list or `go_scattersmith_hoverlabel_font()` object.

`go_scattersmith_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_scattersmith_legendgrouptitle_font()` object.

`go_scattersmith_marker_colorbar_title()`: Sets this color bar's title font. Named list or `go_scattersmith_marker_colorbar_title_font()` object.

namelength — Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to ‘namelength - 3‘ characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow — Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color — go_scattersmith_hoverlabel_font(), go_scattersmith_textfont(): Color: a CSS color string or R color name, or a vector of these.

go_scattersmith_legendgrouptitle_font(), go_scattersmith_marker_colorbar_tickfont(), go_scattersmith_marker_colorbar_title_font(): Color: a CSS color string or R color name.

go_scattersmith_line(): Sets the line color. Color: a CSS color string or R color name.

go_scattersmith_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.cmin‘ and ‘marker.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scattersmith.marker.colorscale, or a vector of these.

go_scattersmith_marker_gradient(): Sets the final color of the gradient fill: the center color for radial, the right for horizontal, or the bottom for vertical. Color: a CSS color string or R color name, or a vector of these.

go_scattersmith_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by scattersmith.marker.line.colorscale, or a vector of these.

go_scattersmith_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name.

go_scattersmith_selected_textfont(): Sets the text font color of selected points. Color: a CSS color string or R color name.

go_scattersmith_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

go_scattersmith_unselected_textfont(): Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

family — go_scattersmith_hoverlabel_font(), go_scattersmith_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String, or a vector of these.

go_scattersmith_legendgrouptitle_font(), go_scattersmith_marker_colorbar_tickfont(), go_scattersmith_marker_colorbar_title_font(): HTML font family - the

typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: `go_scattersmith_hoverlabel_font()`, `go_scattersmith_textfont()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

`go_scattersmith_legendgrouptitle_font()`, `go_scattersmith_marker_colorbar_tickfont()`, `go_scattersmith_marker_colorbar_title_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

shadow
: `go_scattersmith_hoverlabel_font()`, `go_scattersmith_textfont()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

`go_scattersmith_legendgrouptitle_font()`, `go_scattersmith_marker_colorbar_tickfont()`, `go_scattersmith_marker_colorbar_title_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: `go_scattersmith_hoverlabel_font()`, `go_scattersmith_textfont()`: Number in the interval [1, Inf], or a vector of these.

`go_scattersmith_legendgrouptitle_font()`, `go_scattersmith_marker_colorbar_tickfont()`, `go_scattersmith_marker_colorbar_title_font()`: Number in the interval [1, Inf].

`go_scattersmith_marker()`: Sets the marker size (in px). Number in the interval [0, Inf], or a vector of these.

`go_scattersmith_selected_marker()`: Sets the marker size of selected points. Number in the interval [0, Inf].

`go_scattersmith_unselected_marker()`: Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

style
: `go_scattersmith_hoverlabel_font()`, `go_scattersmith_textfont()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`, or a vector of these.

`go_scattersmith_legendgrouptitle_font()`, `go_scattersmith_marker_colorbar_tickfont()`, `go_scattersmith_marker_colorbar_title_font()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

textcase
: `go_scattersmith_hoverlabel_font()`, `go_scattersmith_textfont()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`, or a vector of these.

`go_scattersmith_legendgrouptitle_font()`, `go_scattersmith_marker_colorbar_tickfont()`, `go_scattersmith_marker_colorbar_title_font()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.

variant
: go_scattersmith_hoverlabel_font(), go_scattersmith_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

  go_scattersmith_legendgrouptitle_font(), go_scattersmith_marker_colorbar_tickfont(), go_scattersmith_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_scattersmith_hoverlabel_font(), go_scattersmith_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

  go_scattersmith_legendgrouptitle_font(), go_scattersmith_marker_colorbar_tickfont(), go_scattersmith_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

backoff
: Sets the line back off from the end point of the nth line segment (in px). This option is useful e.g. to avoid overlap with arrowhead markers. With "auto" the lines would trim before markers if ‘marker.angleref‘ is set to "previous". Number in the interval [0, Inf], or a vector of these.

dash
: go_scattersmith_line(): Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\

  go_scattersmith_marker_line(): Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\ or a vector of these.

shape
: Determines the line shape. With "spline" the lines are drawn using spline interpolation. The other available values correspond to step-wise line shapes. One of "linear", "spline".

smoothing
: Has an effect only if ‘shape‘ is set to "spline" Sets the amount of smoothing. 0 corresponds to no smoothing (equivalent to a "linear" shape). Number in the interval [0, 1.3].

width
: go_scattersmith_line(): Sets the line width (in px). Number in the interval [0, Inf].

  go_scattersmith_marker_line(): Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

angle
: Sets the marker angle in respect to ‘angleref‘. Angle in degrees, or a vector of these.

angleref
: Sets the reference for marker angle. With "previous", angle 0 points along the line from the previous point to this one. With "up", angle 0 points toward the top of the screen. One of "previous", "up".

autocolorscale
: go_scattersmith_marker(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be

chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`go_scattersmith_marker_line()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`cauto` `go_scattersmith_marker()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

`go_scattersmith_marker_line()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

`cmax` `go_scattersmith_marker()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

`go_scattersmith_marker_line()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

`cmid` `go_scattersmith_marker()`: Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

`go_scattersmith_marker_line()`: Sets the mid-point of the color domain by scaling ‘marker.line.cmin‘ and/or ‘marker.line.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

`cmin` `go_scattersmith_marker()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`go_scattersmith_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes

are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: "coloraxis", "coloraxis2", "coloraxis3", ...

`colorbar` Named list or `go_scattersmith_marker_colorbar()` object.

`colorscale` `go_scattersmith_marker()`: Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rai
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_scattersmith_marker_line()`: Sets the colorscale. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.line.cmin‘ and ‘marker.line.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,H
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`gradient` Named list or `go_scattersmith_marker_gradient()` object.

`maxdisplayed` Sets a maximum number of points to be drawn on the graph. 0 corresponds to no limit. Number in the interval [0, Inf].

`reversescale` `go_scattersmith_marker()`: Reverses the color mapping if true. Has an effect only if in ‘marker.color‘ is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.

`go_scattersmith_marker_line()`: Reverses the color mapping if true. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. If true, ‘marker.line.cmin‘ will correspond to the last color in the array and ‘marker.line.cmax‘ will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘marker.color‘ is set to a numerical array. Logical.

`sizemin` Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the minimum size (in px) of the rendered marker points. Number in the interval [0, Inf].

`sizemode` Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the rule for which the data in ‘size‘ is converted to pixels. One of `"diameter"`, `"area"`.

`sizeref` Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the scale factor used to determine the rendered size of marker points. Use with ‘sizemin‘ and ‘sizemode‘. Number.

standoff Moves the marker away from the data point in the direction of 'angle' (in px). This can be useful for example if you have another marker at this location and you want to point an arrowhead marker at it. Number in the interval [0, Inf], or a vector of these.

symbol Sets the marker symbol type. Adding 100 is equivalent to appending "-open" to a symbol name. Adding 200 is equivalent to appending "-dot" to a symbol name. Adding 300 is equivalent to appending "-open-dot" or "dot-open" to a symbol name. One of "0", "circle", "100", "circle-open", "200", "circle-dot", "300", "circle-open-dot", "1", "square", "101", "square-open", "201", "square-dot", "301", "square-open-dot", "2", "diamond", "102", "diamond-open", "202", "diamond-dot", "302", "diamond-open-dot", "3", "cross", "103", "cross-open", "203", "cross-dot", "303", "cross-open-dot", "4", "x", "104", "x-open", "204", "x-dot", "304", "x-open-dot", "5", "triangle-up", "105", "triangle-up-open", "205", "triangle-up-dot", "305", "triangle-up-open-dot", "6", "triangle-down", "106", "triangle-down-open", "206", "triangle-down-dot", "306", "triangle-down-open-dot", "7", "triangle-left", "107", "triangle-left-open", "207", "triangle-left-dot", "307", "triangle-left-open-dot", "8", "triangle-right", "108", "triangle-right-open", "208", "triangle-right-dot", "308", "triangle-right-open-do "9", "triangle-ne", "109", "triangle-ne-open", "209", "triangle-ne-dot", "309", "triangle-ne-open-dot", "10", "triangle-se", "110", "triangle-se-open", "210", "triangle-se-dot", "310", "triangle-se-open-dot", "11", "triangle-sw", "111", "triangle-sw-open", "211", "triangle-sw-dot", "311", "triangle-sw-open-dot", "12", "triangle-nw", "112", "triangle-nw-open", "212", "triangle-nw-dot", "312", "triangle-nw-open-dot", "13", "pentagon", "113", "pentagon-open", "213", "pentagon-dot", "313", "pentagon-open-dot", "14", "hexagon", "114", "hexagon-open", "214", "hexagon-dot", "314", "hexagon-open-dot", "15", "hexagon2", "115", "hexagon2-open", "215", "hexagon2-dot", "315", "hexagon2-open-dot", "16", "octagon", "116", "octagon-open", "216", "octagon-dot", "316", "octagon-open-dot", "17", "star", "117", "star-open", "217", "star-dot", "317", "star-open-dot", "18", "hexagram", "118", "hexagram-open", "218", "hexagram-dot", "318", "hexagram-open-dot", "19", "star-triangle-up", "119", "star-triangle-up-open", "219", "star-triangle-up-dot", "319", "star-triangle-up-open-dot", "20", "star-triangle-down", "120", "star-triangle-down-open "220", "star-triangle-down-dot", "320", "star-triangle-down-open-dot", "21", "star-square", "121", "star-square-open", "221", "star-square-dot", "321", "star-square-open-dot", "22", "star-diamond", "122", "star-diamond-open", "222", "star-diamond-dot", "322", "star-diamond-open-dot", "23", "diamond-tall", "123", "diamond-tall-open", "223", "diamond-tall-dot", "323", "diamond-tall-open-dot", "24", "diamond-wide", "124", "diamond-wide-open", "224", "diamond-wide-dot", "324", "diamond-wide-open-dot", "25", "hourglass", "125", "hourglass-open", "26", "bowtie", "126", "bowtie-open", "27", "circle-cross", "127", "circle-cross-open", "28", "circle-x", "128", "circle-x-open", "29", "square-cross", "129", "square-cross-open", "30", "square-x", "130", "square-x-open", "31", "diamond-cross", "131", "diamond-cross-open", "32", "diamond-x", "132", "diamond-x-open", "33", "cross-thin", "133", "cross-thin-open", "34", "x-thin", "134", "x-thin-open", "35", "asterisk", "135", "asterisk-open", "36", "hash", "136", "hash-open", "236", "hash-dot", "336", "hash-open-dot", "37", "y-up", "137", "y-up-open", "38", "y-down", "138", "y-down-open",

"39", "y-left", "139", "y-left-open", "40", "y-right", "140", "y-right-open", "41", "line-ew", "141", "line-ew-open", "42", "line-ns", "142", "line-ns-open", "43", "line-ne", "143", "line-ne-open", "44", "line-nw", "144", "line-nw-open", "45", "arrow-up", "145", "arrow-up-open", "46", "arrow-down", "146", "arrow-down-open", "47", "arrow-left", "147", "arrow-left-open", "48", "arrow-right", "148", "arrow-right-open", "49", "arrow-bar-up", "149", "arrow-bar-up-open", "50", "arrow-bar-down", "150", "arrow-bar-down-open", "51", "arrow-bar-left", "151", "arrow-bar-left-open", "52", "arrow-bar-right", "152", "arrow-bar-right-open", "53", "arrow", "153", "arrow-open", "54", "arrow-wide", "154", "arrow-wide-open", 0, 100, 200, 300, 1, 101, 201, 301, 2, 102, 202, 302, 3, 103, 203, 303, 4, 104, 204, 304, 5, 105, 205, 305, 6, 106, 206, 306, 7, 107, 207, 307, 8, 108, 208, 308, 9, 109, 209, 309, 10, 110, 210, 310, 11, 111, 211, 311, 12, 112, 212, 312, 13, 113, 213, 313, 14, 114, 214, 314, 15, 115, 215, 315, 16, 116, 216, 316, 17, 117, 217, 317, 18, 118, 218, 318, 19, 119, 219, 319, 20, 120, 220, 320, 21, 121, 221, 321, 22, 122, 222, 322, 23, 123, 223, 323, 24, 124, 224, 324, 25, 125, 26, 126, 27, 127, 28, 128, 29, 129, 30, 130, 31, 131, 32, 132, 33, 133, 34, 134, 35, 135, 36, 136, 236, 336, 37, 137, 38, 138, 39, 139, 40, 140, 41, 141, 42, 142, 43, 143, 44, 144, 45, 145, 46, 146, 47, 147, 48, 148, 49, 149, 50, 150, 51, 151, 52, 152, 53, 153, 54, 154, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of "fraction", "pixels".

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar’s tick label font Named list or `go_scattersmith_marker_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops
: List of named lists or go_scattersmith_marker_colorbar_tickformatstop() objects.

ticklabeloverflow
: Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelstep
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

ticklen
: Sets the tick length (in px). Number in the interval [0, Inf].

tickmode
: Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of "auto", "linear", "array".

tickprefix
: Sets a tick label prefix. String.

ticks
: Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of "outside", "inside", "".

ticksuffix
: Sets a tick label suffix. String.

ticktext
: Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals
: Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth
: Sets the tick width (in px). Number in the interval [0, Inf].

title
: String (the title text), named list or go_scattersmith_marker_colorbar_title() object.

x
: Sets the x position with respect to 'xref' of the color bar (in plot fraction). When 'xref' is "paper", defaults to 1.02 when 'orientation' is "v" and 0.5 when 'orientation' is "h". When 'xref' is "container", defaults to 1 when 'orientation' is "v" and 0.5 when 'orientation' is "h". Must be between 0 and 1 if 'xref' is "container" and between "-2" and 3 if 'xref' is "paper". Number.

xanchor — Sets this color bar's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the color bar. Defaults to "left" when 'orientation' is "v" and "center" when 'orientation' is "h". One of "left", "center", "right".

xpad — Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container 'x' refers to. "container" spans the entire 'width' of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".

y — Sets the y position with respect to 'yref' of the color bar (in plot fraction). When 'yref' is "paper", defaults to 0.5 when 'orientation' is "v" and 1.02 when 'orientation' is "h". When 'yref' is "container", defaults to 0.5 when 'orientation' is "v" and 1 when 'orientation' is "h". Must be between 0 and 1 if 'yref' is "container" and between "-2" and 3 if 'yref' is "paper". Number.

yanchor — Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

side — Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of "right", "top", "bottom".

type — Sets the type of gradient used to fill the markers One of "radial", "horizontal", "vertical", "none", or a vector of these.

Details

go_scattersmith() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_scattersmith() creates an object and adds it to a figure.

Nested objects documented here:

`go_scattersmith_hoverlabel()` `scattersmith.hoverlabel`

`go_scattersmith_hoverlabel_font()` `scattersmith.hoverlabel.font`: Sets the font used in hover labels.

`go_scattersmith_legendgrouptitle()` `scattersmith.legendgrouptitle`

`go_scattersmith_legendgrouptitle_font()` `scattersmith.legendgrouptitle.font`: Sets this legend group's title font.

`go_scattersmith_line()` `scattersmith.line`

`go_scattersmith_marker()` `scattersmith.marker`

`go_scattersmith_marker_colorbar()` `scattersmith.marker.colorbar`

`go_scattersmith_marker_colorbar_tickfont()` `scattersmith.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_scattersmith_marker_colorbar_tickformatstop()` `scattersmith.marker.colorbar.tickformatstop`

`go_scattersmith_marker_colorbar_title()` `scattersmith.marker.colorbar.title`

`go_scattersmith_marker_colorbar_title_font()` `scattersmith.marker.colorbar.title.font`: Sets this color bar's title font.

`go_scattersmith_marker_gradient()` `scattersmith.marker.gradient`

`go_scattersmith_marker_line()` `scattersmith.marker.line`

`go_scattersmith_selected()` `scattersmith.selected`

`go_scattersmith_selected_marker()` `scattersmith.selected.marker`

`go_scattersmith_selected_textfont()` `scattersmith.selected.textfont`

`go_scattersmith_textfont()` `scattersmith.textfont`: Sets the text font.

`go_scattersmith_unselected()` `scattersmith.unselected`

`go_scattersmith_unselected_marker()` `scattersmith.unselected.marker`

`go_scattersmith_unselected_textfont()` `scattersmith.unselected.textfont`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_scattersmith(mode = "markers", marker_color = "red")
```

`add_scatterternary` *Scatterternary traces*

Description

Provides similar functionality to the "scatter" type but on a ternary phase diagram. The data is provided by at least two arrays out of ‘a‘, ‘b‘, ‘c‘ triplets.

Usage

```
add_scatterternary(fig, ...)

## Default S3 method:
add_scatterternary(
  fig,
  ...,
  data = NULL,
  a = NULL,
  b = NULL,
  c = NULL,
  cliponaxis = NULL,
  connectgaps = NULL,
  customdata = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  subplot = NULL,
```

```
  sum = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_scatterternary(
  arg = NULL,
  ...,
  a = NULL,
  b = NULL,
  c = NULL,
  cliponaxis = NULL,
  connectgaps = NULL,
  customdata = NULL,
  fill = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meta = NULL,
  mode = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  subplot = NULL,
```

```
  sum = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_scatterternary_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_scatterternary_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterternary_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)
```

```
  go_scatterternary_legendgrouptitle_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_scatterternary_line(
    arg = NULL,
    ...,
    backoff = NULL,
    color = NULL,
    dash = NULL,
    shape = NULL,
    smoothing = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_scatterternary_marker(
    arg = NULL,
    ...,
    angle = NULL,
    angleref = NULL,
    autocolorscale = NULL,
    cauto = NULL,
    cmax = NULL,
    cmid = NULL,
    cmin = NULL,
    color = NULL,
    coloraxis = NULL,
    colorbar = NULL,
    colorscale = NULL,
    gradient = NULL,
    line = NULL,
    maxdisplayed = NULL,
    opacity = NULL,
    reversescale = NULL,
    showscale = NULL,
    size = NULL,
```

```
  sizemin = NULL,
  sizemode = NULL,
  sizeref = NULL,
  standoff = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_scatterternary_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
```

```
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_scatterternary_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterternary_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_scatterternary_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_scatterternary_marker_colorbar_title_font(
```

```
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterternary_marker_gradient(
  arg = NULL,
  ...,
  color = NULL,
  type = NULL,
  skip_invalid = FALSE
)

go_scatterternary_marker_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  dash = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_scatterternary_selected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scatterternary_selected_marker(
```

```
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scatterternary_selected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

go_scatterternary_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_scatterternary_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_scatterternary_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_scatterternary_unselected_textfont(
```

```
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
a	Sets the quantity of component ‘a‘ in each data point. If ‘a‘, ‘b‘, and ‘c‘ are all provided, they need not be normalized, only the relative values matter. If only two arrays are provided they must be normalized to match ‘ternary<i>.sum‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
b	Sets the quantity of component ‘a‘ in each data point. If ‘a‘, ‘b‘, and ‘c‘ are all provided, they need not be normalized, only the relative values matter. If only two arrays are provided they must be normalized to match ‘ternary<i>.sum‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
c	Sets the quantity of component ‘a‘ in each data point. If ‘a‘, ‘b‘, and ‘c‘ are all provided, they need not be normalized, only the relative values matter. If only two arrays are provided they must be normalized to match ‘ternary<i>.sum‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
cliponaxis	Determines whether or not markers and text nodes are clipped about the subplot axes. To show markers and text nodes above axis lines and tick labels, make sure to set ‘xaxis.layer‘ and ‘yaxis.layer‘ to "below traces". Logical.
connectgaps	Determines whether or not gaps (i.e. {nan} or missing values) in the provided data arrays are connected. Logical.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
fill	Sets the area to fill with a solid color. Use with ‘fillcolor‘ if not "none". scatterternary has a subset of the options available to scatter. "toself" connects the endpoints of the trace (or each segment of the trace if it has gaps) into a closed shape. "tonext" fills the space between two traces if one completely encloses the other (eg consecutive contour lines), and behaves like "toself" if there is no trace before it. "tonext" should not be used if one trace does not enclose the other. One of "none", "toself", "tonext".
fillcolor	Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name.

`hoverinfo` Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of `"a"`, `"b"`, `"c"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_scatterternary_hoverlabel()` object.

`hoveron` Do the hover effects highlight individual points (markers or line points) or do they highlight filled regions? If the fill is "toself" or "tonext" and there are no markers or text, then the default is "fills", otherwise it is "points". Any combination of `"points"`, `"fills"` joined with `"+"`.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example ‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

`hovertemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Sets hover text elements associated with each (a,b,c) point. If a single string, the same string appears over all the data points. If an array of strings, the items are mapped in order to the data points in (a,b,c). To be seen, trace ‘hoverinfo‘ must contain a "text" flag. String, or a vector of these.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_scatterternary_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line`
: `go_scatterternary()`: Named list or `go_scatterternary_line()` object.
`go_scatterternary_marker()`: Named list or `go_scatterternary_marker_line()` object.

`marker`
: `go_scatterternary()`: Named list or `go_scatterternary_marker()` object.
`go_scatterternary_selected()`: Named list or `go_scatterternary_selected_marker()` object.
`go_scatterternary_unselected()`: Named list or `go_scatterternary_unselected_marker()` object.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`mode`
: Determines the drawing mode for this scatter trace. If the provided ‘mode‘ includes "text" then the ‘text‘ elements appear at the coordinates. Otherwise, the ‘text‘ elements appear on hover. If there are less than 20 points and the trace is not stacked then the default is "lines+markers". Otherwise, "lines". Any combination of `"lines"`, `"markers"`, `"text"` joined with `"+"`, or one of `"none"`.

`name`
: `go_scatterternary()`: Sets the trace name. The trace name appears as the legend item and on hover. String.
`go_scatterternary_marker_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity`
: `go_scatterternary()`: Sets the opacity of the trace. Number in the interval [0, 1].
`go_scatterternary_marker()`: Sets the marker opacity. Number in the interval [0, 1], or a vector of these.
`go_scatterternary_selected_marker()`: Sets the marker opacity of selected points. Number in the interval [0, 1].

	`go_scatterternary_unselected_marker()`: Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].
`selected`	Named list or `go_scatterternary_selected()` object.
`selectedpoints`	Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the 'unselected' are turned on for all points, whereas, any other non-array values means no selection all where the 'selected' and 'unselected' styles have no effect. Any value.
`showlegend`	Determines whether or not an item corresponding to this trace is shown in the legend. Logical.
`subplot`	Sets a reference between this trace's data coordinates and a ternary subplot. If "ternary" (the default value), the data refer to 'layout.ternary'. If "ternary2", the data refer to 'layout.ternary2', and so on. Subplot id: `"ternary"`, `"ternary2"`, `"ternary3"`, ...
`sum`	The number each triplet should sum to, if only two of 'a', 'b', and 'c' are provided. This overrides 'ternary<i>.sum' to normalize this specific trace, but does not affect the values displayed on the axes. 0 (or missing) means to use ternary<i>.sum Number in the interval [0, Inf].
`text`	`go_scatterternary()`: Sets text elements associated with each (a,b,c) point. If a single string, the same string appears over all the data points. If an array of strings, the items are mapped in order to the data points in (a,b,c). If trace 'hoverinfo' contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these. `go_scatterternary_legendgrouptitle()`: Sets the title of the legend group. String. `go_scatterternary_marker_colorbar_title()`: Sets the title of the color bar. String.
`textfont`	`go_scatterternary()`: Sets the text font. Named list or `go_scatterternary_textfont()` object. `go_scatterternary_selected()`: Named list or `go_scatterternary_selected_textfont()` object. `go_scatterternary_unselected()`: Named list or `go_scatterternary_unselected_textfont()` object.
`textposition`	Sets the positions of the 'text' elements with respects to the (x,y) coordinates. One of `"top left"`, `"top center"`, `"top right"`, `"middle left"`, `"middle center"`, `"middle right"`, `"bottom left"`, `"bottom center"`, `"bottom right"`, or a vector of these.
`texttemplate`	Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format

for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'a', 'b', 'c' and 'text'. String, or a vector of these.

`texttemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`uid`
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision`
: Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

`unselected`
: Named list or `go_scatterternary_unselected()` object.

`visible`
: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`row`
: Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`col`
: Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`skip_invalid`
: If `TRUE`, silently drop invalid properties instead of raising an error.

`arg`
: Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

`align`
: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`bgcolor`
: `go_scatterternary_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`go_scatterternary_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

`bordercolor`
: `go_scatterternary_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

	`go_scatterternary_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.
`font`	`go_scatterternary_hoverlabel()`: Sets the font used in hover labels. Named list or `go_scatterternary_hoverlabel_font()` object. `go_scatterternary_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_scatterternary_legendgrouptitle_font()` object. `go_scatterternary_marker_colorbar_title()`: Sets this color bar's title font. Named list or `go_scatterternary_marker_colorbar_title_font()` object.
`namelength`	Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.
`showarrow`	Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.
`color`	`go_scatterternary_hoverlabel_font()`, `go_scatterternary_textfont()`: Color: a CSS color string or R color name, or a vector of these. `go_scatterternary_legendgrouptitle_font()`, `go_scatterternary_marker_colorbar_tickfont` `go_scatterternary_marker_colorbar_title_font()`: Color: a CSS color string or R color name. `go_scatterternary_line()`: Sets the line color. Color: a CSS color string or R color name. `go_scatterternary_marker()`: Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.cmin' and 'marker.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by `scatterternary.marker.colorscale`, or a vector of these. `go_scatterternary_marker_gradient()`: Sets the final color of the gradient fill: the center color for radial, the right for horizontal, or the bottom for vertical. Color: a CSS color string or R color name, or a vector of these. `go_scatterternary_marker_line()`: Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.line.cmin' and 'marker.line.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by `scatterternary.marker.line.colorscale`, or a vector of these. `go_scatterternary_selected_marker()`: Sets the marker color of selected points. Color: a CSS color string or R color name. `go_scatterternary_selected_textfont()`: Sets the text font color of selected points. Color: a CSS color string or R color name. `go_scatterternary_unselected_marker()`: Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

go_scatterternary_unselected_textfont(): Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

family

go_scatterternary_hoverlabel_font(), go_scatterternary_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

go_scatterternary_legendgrouptitle_font(), go_scatterternary_marker_colorbar_tickfont
go_scatterternary_marker_colorbar_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition

go_scatterternary_hoverlabel_font(), go_scatterternary_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_scatterternary_legendgrouptitle_font(), go_scatterternary_marker_colorbar_tickfont
go_scatterternary_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow

go_scatterternary_hoverlabel_font(), go_scatterternary_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_scatterternary_legendgrouptitle_font(), go_scatterternary_marker_colorbar_tickfont
go_scatterternary_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size

go_scatterternary_hoverlabel_font(), go_scatterternary_textfont(): Number in the interval [1, Inf], or a vector of these.

go_scatterternary_legendgrouptitle_font(), go_scatterternary_marker_colorbar_tickfont
go_scatterternary_marker_colorbar_title_font(): Number in the interval [1, Inf].

go_scatterternary_marker(): Sets the marker size (in px). Number in the interval [0, Inf], or a vector of these.

go_scatterternary_selected_marker(): Sets the marker size of selected points. Number in the interval [0, Inf].

go_scatterternary_unselected_marker(): Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

style — go_scatterternary_hoverlabel_font(), go_scatterternary_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_scatterternary_legendgrouptitle_font(), go_scatterternary_marker_colorbar_tickfont go_scatterternary_marker_colorbar_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase — go_scatterternary_hoverlabel_font(), go_scatterternary_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_scatterternary_legendgrouptitle_font(), go_scatterternary_marker_colorbar_tickfont go_scatterternary_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant — go_scatterternary_hoverlabel_font(), go_scatterternary_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_scatterternary_legendgrouptitle_font(), go_scatterternary_marker_colorbar_tickfont go_scatterternary_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight — go_scatterternary_hoverlabel_font(), go_scatterternary_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_scatterternary_legendgrouptitle_font(), go_scatterternary_marker_colorbar_tickfont go_scatterternary_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

backoff — Sets the line back off from the end point of the nth line segment (in px). This option is useful e.g. to avoid overlap with arrowhead markers. With "auto" the lines would trim before markers if ‘marker.angleref‘ is set to "previous". Number in the interval [0, Inf], or a vector of these.

dash — go_scatterternary_line(): Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\

go_scatterternary_marker_line(): Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$", or a vector of these.

shape — Determines the line shape. With "spline" the lines are drawn using spline interpolation. The other available values correspond to step-wise line shapes. One of "linear", "spline".

`smoothing` Has an effect only if ‘shape‘ is set to "spline" Sets the amount of smoothing. 0 corresponds to no smoothing (equivalent to a "linear" shape). Number in the interval [0, 1.3].

`width` `go_scatterternary_line()`: Sets the line width (in px). Number in the interval [0, Inf].

`go_scatterternary_marker_line()`: Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

`angle` Sets the marker angle in respect to ‘angleref‘. Angle in degrees, or a vector of these.

`angleref` Sets the reference for marker angle. With "previous", angle 0 points along the line from the previous point to this one. With "up", angle 0 points toward the top of the screen. One of `"previous"`, `"up"`.

`autocolorscale` `go_scatterternary_marker()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`go_scatterternary_marker_line()`: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`cauto` `go_scatterternary_marker()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

`go_scatterternary_marker_line()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

`cmax` `go_scatterternary_marker()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

`go_scatterternary_marker_line()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

`cmid` `go_scatterternary_marker()`: Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have

the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

`go_scatterternary_marker_line()`: Sets the mid-point of the color domain by scaling ‘marker.line.cmin‘ and/or ‘marker.line.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

`cmin` — `go_scatterternary_marker()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`go_scatterternary_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

`coloraxis` — Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` — Named list or `go_scatterternary_marker_colorbar()` object.

`colorscale` — `go_scatterternary_marker()`: Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rai
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_scatterternary_marker_line()`: Sets the colorscale. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.line.cmin‘ and ‘marker.line.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,H
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`gradient` — Named list or `go_scatterternary_marker_gradient()` object.

`maxdisplayed` — Sets a maximum number of points to be drawn on the graph. 0 corresponds to no limit. Number in the interval [0, Inf].

reversescale
: go_scatterternary_marker(): Reverses the color mapping if true. Has an effect only if in ‘marker.color‘ is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color.

go_scatterternary_marker_line(): Reverses the color mapping if true. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. If true, ‘marker.line.cmin‘ will correspond to the last color in the array and ‘marker.line.cmax‘ will correspond to the first color. Logical.

showscale
: Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘marker.color‘ is set to a numerical array. Logical.

sizemin
: Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the minimum size (in px) of the rendered marker points. Number in the interval [0, Inf].

sizemode
: Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the rule for which the data in ‘size‘ is converted to pixels. One of "diameter", "area".

sizeref
: Has an effect only if ‘marker.size‘ is set to a numerical array. Sets the scale factor used to determine the rendered size of marker points. Use with ‘sizemin‘ and ‘sizemode‘. Number.

standoff
: Moves the marker away from the data point in the direction of ‘angle‘ (in px). This can be useful for example if you have another marker at this location and you want to point an arrowhead marker at it. Number in the interval [0, Inf], or a vector of these.

symbol
: Sets the marker symbol type. Adding 100 is equivalent to appending "-open" to a symbol name. Adding 200 is equivalent to appending "-dot" to a symbol name. Adding 300 is equivalent to appending "-open-dot" or "dot-open" to a symbol name. One of "0", "circle", "100", "circle-open", "200", "circle-dot", "300", "circle-open-dot", "1", "square", "101", "square-open", "201", "square-dot", "301", "square-open-dot", "2", "diamond", "102", "diamond-open", "202", "diamond-dot", "302", "diamond-open-dot", "3", "cross", "103", "cross-open", "203", "cross-dot", "303", "cross-open-dot", "4", "x", "104", "x-open", "204", "x-dot", "304", "x-open-dot", "5", "triangle-up", "105", "triangle-up-open", "205", "triangle-up-dot", "305", "triangle-up-open-dot", "6", "triangle-down", "106", "triangle-down-open", "206", "triangle-down-dot", "306", "triangle-down-open-dot", "7", "triangle-left", "107", "triangle-left-open", "207", "triangle-left-dot", "307", "triangle-left-open-dot", "8", "triangle-right", "108", "triangle-right-open", "208", "triangle-right-dot", "308", "triangle-right-open-do "9", "triangle-ne", "109", "triangle-ne-open", "209", "triangle-ne-dot", "309", "triangle-ne-open-dot", "10", "triangle-se", "110", "triangle-se-open", "210", "triangle-se-dot", "310", "triangle-se-open-dot", "11", "triangle-sw", "111", "triangle-sw-open", "211", "triangle-sw-dot", "311", "triangle-sw-open-dot", "12", "triangle-nw", "112", "triangle-nw-open", "212", "triangle-nw-dot", "312", "triangle-nw-open-dot", "13", "pentagon", "113", "pentagon-open", "213", "pentagon-dot", "313", "pentagon-open-dot", "14", "hexagon", "114", "hexagon-open", "214", "hexagon-dot", "314", "hexagon-open-dot", "15", "hexagon2", "115", "hexagon2-open", "215", "hexagon2-dot", "315", "hexagon2-open-dot", "16", "octagon", "116", "octagon-open", "216", "octagon-dot", "316", "octagon-open-dot", "17", "star", "117", "star-open", "217", "star-dot", "317", "star-open-dot", "18", "hexagram", "118", "hexagram-open",

"218", "hexagram-dot", "318", "hexagram-open-dot", "19", "star-triangle-up",
"119", "star-triangle-up-open", "219", "star-triangle-up-dot", "319",
"star-triangle-up-open-dot", "20", "star-triangle-down", "120", "star-triangle-down-open
"220", "star-triangle-down-dot", "320", "star-triangle-down-open-dot",
"21", "star-square", "121", "star-square-open", "221", "star-square-dot",
"321", "star-square-open-dot", "22", "star-diamond", "122", "star-diamond-open",
"222", "star-diamond-dot", "322", "star-diamond-open-dot", "23", "diamond-tall",
"123", "diamond-tall-open", "223", "diamond-tall-dot", "323", "diamond-tall-open-dot",
"24", "diamond-wide", "124", "diamond-wide-open", "224", "diamond-wide-dot",
"324", "diamond-wide-open-dot", "25", "hourglass", "125", "hourglass-open",
"26", "bowtie", "126", "bowtie-open", "27", "circle-cross", "127", "circle-cross-open",
"28", "circle-x", "128", "circle-x-open", "29", "square-cross", "129",
"square-cross-open", "30", "square-x", "130", "square-x-open", "31",
"diamond-cross", "131", "diamond-cross-open", "32", "diamond-x", "132",
"diamond-x-open", "33", "cross-thin", "133", "cross-thin-open", "34",
"x-thin", "134", "x-thin-open", "35", "asterisk", "135", "asterisk-open",
"36", "hash", "136", "hash-open", "236", "hash-dot", "336", "hash-open-dot",
"37", "y-up", "137", "y-up-open", "38", "y-down", "138", "y-down-open",
"39", "y-left", "139", "y-left-open", "40", "y-right", "140", "y-right-open",
"41", "line-ew", "141", "line-ew-open", "42", "line-ns", "142", "line-ns-open",
"43", "line-ne", "143", "line-ne-open", "44", "line-nw", "144", "line-nw-open",
"45", "arrow-up", "145", "arrow-up-open", "46", "arrow-down", "146",
"arrow-down-open", "47", "arrow-left", "147", "arrow-left-open", "48",
"arrow-right", "148", "arrow-right-open", "49", "arrow-bar-up", "149",
"arrow-bar-up-open", "50", "arrow-bar-down", "150", "arrow-bar-down-open",
"51", "arrow-bar-left", "151", "arrow-bar-left-open", "52", "arrow-bar-right",
"152", "arrow-bar-right-open", "53", "arrow", "153", "arrow-open", "54",
"arrow-wide", "154", "arrow-wide-open", 0, 100, 200, 300, 1, 101, 201, 301,
2, 102, 202, 302, 3, 103, 203, 303, 4, 104, 204, 304, 5, 105, 205, 305, 6, 106,
206, 306, 7, 107, 207, 307, 8, 108, 208, 308, 9, 109, 209, 309, 10, 110, 210, 310,
11, 111, 211, 311, 12, 112, 212, 312, 13, 113, 213, 313, 14, 114, 214, 314, 15,
115, 215, 315, 16, 116, 216, 316, 17, 117, 217, 317, 18, 118, 218, 318, 19, 119,
219, 319, 20, 120, 220, 320, 21, 121, 221, 321, 22, 122, 222, 322, 23, 123, 223,
323, 24, 124, 224, 324, 25, 125, 26, 126, 27, 127, 28, 128, 29, 129, 30, 130, 31,
131, 32, 132, 33, 133, 34, 134, 35, 135, 36, 136, 236, 336, 37, 137, 38, 138, 39,
139, 40, 140, 41, 141, 42, 142, 43, 143, 44, 144, 45, 145, 46, 146, 47, 147, 48,
148, 49, 149, 50, 150, 51, 151, 52, 152, 53, 153, 54, 154, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30).

If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

labelalias Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

len Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

lenmode Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of "fraction", "pixels".

minexponent Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

nticks Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

orientation Sets the orientation of the colorbar. One of "h", "v".

outlinecolor Sets the axis line color. Color: a CSS color string or R color name.

outlinewidth Sets the width (in px) of the axis line. Number in the interval [0, Inf].

separatethousands
If "true", even 4-digit integers are separated Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".

showticksuffix Same as 'showtickprefix' but for tick suffixes. One of "all", "first", "last", "none".

thickness Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of "fraction", "pixels".

tick0 Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick'

for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_scatterternary_marker_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops`
List of named lists or `go_scatterternary_marker_colorbar_tickformatstop()` objects.

`ticklabeloverflow`
Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition`
Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

ticks	Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of "outside", "inside", "".
ticksuffix	Sets a tick label suffix. String.
ticktext	Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
tickvals	Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
tickwidth	Sets the tick width (in px). Number in the interval [0, Inf].
title	String (the title text), named list or go_scatterternary_marker_colorbar_title() object.
x	Sets the x position with respect to 'xref' of the color bar (in plot fraction). When 'xref' is "paper", defaults to 1.02 when 'orientation' is "v" and 0.5 when 'orientation' is "h". When 'xref' is "container", defaults to 1 when 'orientation' is "v" and 0.5 when 'orientation' is "h". Must be between 0 and 1 if 'xref' is "container" and between "-2" and 3 if 'xref' is "paper". Number.
xanchor	Sets this color bar's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the color bar. Defaults to "left" when 'orientation' is "v" and "center" when 'orientation' is "h". One of "left", "center", "right".
xpad	Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].
xref	Sets the container 'x' refers to. "container" spans the entire 'width' of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".
y	Sets the y position with respect to 'yref' of the color bar (in plot fraction). When 'yref' is "paper", defaults to 0.5 when 'orientation' is "v" and 1.02 when 'orientation' is "h". When 'yref' is "container", defaults to 0.5 when 'orientation' is "v" and 1 when 'orientation' is "h". Must be between 0 and 1 if 'yref' is "container" and between "-2" and 3 if 'yref' is "paper". Number.
yanchor	Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of "top", "middle", "bottom".
ypad	Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].
yref	Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".
dtickrange	range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

`enabled` Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

`templateitemname`
Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

`value` string - dtickformat for described zoom level, the same as "tickformat" String.

`side` Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of `"right"`, `"top"`, `"bottom"`.

`type` Sets the type of gradient used to fill the markers One of `"radial"`, `"horizontal"`, `"vertical"`, `"none"`, or a vector of these.

Details

`go_scatterternary()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_scatterternary()` creates an object and adds it to a figure.

Nested objects documented here:

`go_scatterternary_hoverlabel()` `scatterternary.hoverlabel`

`go_scatterternary_hoverlabel_font()` `scatterternary.hoverlabel.font`: Sets the font used in hover labels.

`go_scatterternary_legendgrouptitle()` `scatterternary.legendgrouptitle`

`go_scatterternary_legendgrouptitle_font()` `scatterternary.legendgrouptitle.font`: Sets this legend group's title font.

`go_scatterternary_line()` `scatterternary.line`

`go_scatterternary_marker()` `scatterternary.marker`

`go_scatterternary_marker_colorbar()` `scatterternary.marker.colorbar`

`go_scatterternary_marker_colorbar_tickfont()` `scatterternary.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_scatterternary_marker_colorbar_tickformatstop()` `scatterternary.marker.colorbar.tickformatstop`

`go_scatterternary_marker_colorbar_title()` `scatterternary.marker.colorbar.title`

`go_scatterternary_marker_colorbar_title_font()` `scatterternary.marker.colorbar.title.font`: Sets this color bar's title font.

`go_scatterternary_marker_gradient()` `scatterternary.marker.gradient`

`go_scatterternary_marker_line()` `scatterternary.marker.line`

`go_scatterternary_selected()` `scatterternary.selected`

`go_scatterternary_selected_marker()` `scatterternary.selected.marker`

`go_scatterternary_selected_textfont()` `scatterternary.selected.textfont`

`go_scatterternary_textfont()` `scatterternary.textfont`: Sets the text font.

`go_scatterternary_unselected()` `scatterternary.unselected`

`go_scatterternary_unselected_marker()` `scatterternary.unselected.marker`

`go_scatterternary_unselected_textfont()` `scatterternary.unselected.textfont`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_scatterternary(mode = "markers", marker_color = "red")
```

---

add_selection *Layout selection objects*

Description

Properties of `layout.selection`.

Usage

```
add_selection(
  fig,
  arg = NULL,
  ...,
  line = NULL,
  name = NULL,
  opacity = NULL,
  path = NULL,
  templateitemname = NULL,
  type = NULL,
  x0 = NULL,
  x1 = NULL,
  xref = NULL,
  y0 = NULL,
  y1 = NULL,
  yref = NULL,
  row = NULL,
  col = NULL,
```

```
  secondary_y = NULL,
  exclude_empty_subplots = FALSE,
  skip_invalid = FALSE
)

go_layout_selection(
  arg = NULL,
  ...,
  line = NULL,
  name = NULL,
  opacity = NULL,
  path = NULL,
  templateitemname = NULL,
  type = NULL,
  x0 = NULL,
  x1 = NULL,
  xref = NULL,
  y0 = NULL,
  y1 = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_layout_selection_line(
  arg = NULL,
  ...,
  color = NULL,
  dash = NULL,
  width = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
line	Named list or go_layout_selection_line() object.
name	When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.
opacity	Sets the opacity of the selection. Number in the interval [0, 1].

path — For ‘type‘ "path" - a valid SVG path similar to ‘shapes.path‘ in data coordinates. Allowed segments are: M, L and Z. String.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

type — Specifies the selection type to be drawn. If "rect", a rectangle is drawn linking (‘x0‘,‘y0‘), (‘x1‘,‘y0‘), (‘x1‘,‘y1‘) and (‘x0‘,‘y1‘). If "path", draw a custom SVG path using ‘path‘. One of `"rect"`, `"path"`.

x0 — Sets the selection’s starting x position. Any value.

x1 — Sets the selection’s end x position. Any value.

xref — Sets the selection’s x coordinate axis. If set to a x axis id (e.g. "x" or "x2"), the ‘x‘ position refers to a x coordinate. If set to "paper", the ‘x‘ position refers to the distance from the left of the plotting area in normalized coordinates where 0 (1) corresponds to the left (right). If set to a x axis ID followed by "domain" (separated by a space), the position behaves like for "paper", but refers to the distance in fractions of the domain length from the left of the domain of that axis: e.g., "x2 domain" refers to the domain of the second x axis and a x position of 0.5 refers to the point between the left and the right of the domain of the second x axis. One of `"paper"`, or a string matching `"^x([2-9]|[1-9][0-9]+)?(domain)?$"`.

y0 — Sets the selection’s starting y position. Any value.

y1 — Sets the selection’s end y position. Any value.

yref — Sets the selection’s x coordinate axis. If set to a y axis id (e.g. "y" or "y2"), the ‘y‘ position refers to a y coordinate. If set to "paper", the ‘y‘ position refers to the distance from the bottom of the plotting area in normalized coordinates where 0 (1) corresponds to the bottom (top). If set to a y axis ID followed by "domain" (separated by a space), the position behaves like for "paper", but refers to the distance in fractions of the domain length from the bottom of the domain of that axis: e.g., "y2 domain" refers to the domain of the second y axis and a y position of 0.5 refers to the point between the bottom and the top of the domain of the second y axis. One of `"paper"`, or a string matching `"^y([2-9]|[1-9][0-9]+)?(domain)?$"`.

row — Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

col — Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

secondary_y — If `TRUE`, use the secondary y axis of the subplot.

exclude_empty_subplots — If `TRUE`, skip subplots that have no traces or layout objects.

skip_invalid — If `TRUE`, silently drop invalid properties instead of raising an error.

color — Sets the line color. Color: a CSS color string or R color name.

dash	Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.
width	Sets the line width (in px). Number in the interval [1, Inf].

Details

`go_layout_selection()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_selection()` creates an object and adds it to a figure.

Nested objects documented here:

`go_layout_selection_line()` `layout.selection.line`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

---

add_shape *Layout shape objects*

Description

Properties of `layout.shape`.

Usage

```
add_shape(
  fig,
  arg = NULL,
  ...,
  editable = NULL,
  fillcolor = NULL,
  fillrule = NULL,
  label = NULL,
  layer = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
```

```
  line = NULL,
  name = NULL,
  opacity = NULL,
  path = NULL,
  showlegend = NULL,
  templateitemname = NULL,
  type = NULL,
  visible = NULL,
  x0 = NULL,
  x0shift = NULL,
  x1 = NULL,
  x1shift = NULL,
  xanchor = NULL,
  xref = NULL,
  xsizemode = NULL,
  y0 = NULL,
  y0shift = NULL,
  y1 = NULL,
  y1shift = NULL,
  yanchor = NULL,
  yref = NULL,
  ysizemode = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  exclude_empty_subplots = FALSE,
  skip_invalid = FALSE
)

go_layout_shape(
  arg = NULL,
  ...,
  editable = NULL,
  fillcolor = NULL,
  fillrule = NULL,
  label = NULL,
  layer = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  name = NULL,
  opacity = NULL,
  path = NULL,
  showlegend = NULL,
  templateitemname = NULL,
```

```
  type = NULL,
  visible = NULL,
  x0 = NULL,
  x0shift = NULL,
  x1 = NULL,
  x1shift = NULL,
  xanchor = NULL,
  xref = NULL,
  xsizemode = NULL,
  y0 = NULL,
  y0shift = NULL,
  y1 = NULL,
  y1shift = NULL,
  yanchor = NULL,
  yref = NULL,
  ysizemode = NULL,
  skip_invalid = FALSE
)

go_layout_shape_label(
  arg = NULL,
  ...,
  font = NULL,
  padding = NULL,
  text = NULL,
  textangle = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  xanchor = NULL,
  yanchor = NULL,
  skip_invalid = FALSE
)

go_layout_shape_label_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

```
go_layout_shape_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_layout_shape_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_shape_line(
  arg = NULL,
  ...,
  color = NULL,
  dash = NULL,
  width = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
editable	Determines whether the shape could be activated for edit or not. Has no effect when the older editable shapes mode is enabled via 'config.editable' or 'config.edits.shapePosition'. Logical.
fillcolor	Sets the color filling the shape's interior. Only applies to closed shapes. Color: a CSS color string or R color name.
fillrule	Determines which regions of complex paths constitute the interior. For more

info please visit https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/fill-rule One of `"evenodd"`, `"nonzero"`.

`label` Named list or `go_layout_shape_label()` object.

`layer` Specifies whether shapes are drawn below gridlines ("below"), between gridlines and traces ("between") or above traces ("above"). One of `"below"`, `"above"`, `"between"`.

`legend` Sets the reference to a legend to show this shape in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this shape. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
Named list or `go_layout_shape_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this shape. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this shape. Number in the interval [0, Inf].

`line` Named list or `go_layout_shape_line()` object.

`name` When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` Sets the opacity of the shape. Number in the interval [0, 1].

`path` For ‘type‘ "path" - a valid SVG path with the pixel values replaced by data values in ‘xsizemode‘/‘ysizemode‘ being "scaled" and taken unmodified as pixels relative to ‘xanchor‘ and ‘yanchor‘ in case of "pixel" size mode. There are a few restrictions / quirks only absolute instructions, not relative. So the allowed segments are: M, L, H, V, Q, C, T, S, and Z arcs (A) are not allowed because radius rx and ry are relative. In the future we could consider supporting relative commands, but we would have to decide on how to handle date and log axes. Note that even as is, Q and C Bezier paths that are smooth on linear axes may not be smooth on log, and vice versa. no chained "polybezier" commands - specify the segment type for each one. On category axes, values are numbers scaled to the serial numbers of categories because using the categories themselves there would be no way to describe fractional positions On data axes: because space and T are both normal components of path strings, we can’t use either to separate date from time parts. Therefore we’ll use underscore for this purpose: 2015-02-21_13:45:56.789 String.

showlegend Determines whether or not this shape is shown in the legend. Logical.

templateitemname
Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

type Specifies the shape type to be drawn. If "line", a line is drawn from (‘x0‘,‘y0‘) to (‘x1‘,‘y1‘) with respect to the axes’ sizing mode. If "circle", a circle is drawn from ((‘x0‘+‘x1‘)/2, (‘y0‘+‘y1‘)/2)) with radius (|(‘x0‘+‘x1‘)/2 - ‘x0‘|, |(‘y0‘+‘y1‘)/2 -‘y0‘)|) with respect to the axes’ sizing mode. If "rect", a rectangle is drawn linking (‘x0‘,‘y0‘), (‘x1‘,‘y0‘), (‘x1‘,‘y1‘), (‘x0‘,‘y1‘), (‘x0‘,‘y0‘) with respect to the axes’ sizing mode. If "path", draw a custom SVG path using ‘path‘. with respect to the axes’ sizing mode. One of `"circle"`, `"rect"`, `"path"`, `"line"`.

visible Determines whether or not this shape is visible. If "legendonly", the shape is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

x0 Sets the shape’s starting x position. See ‘type‘ and ‘xsizemode‘ for more info. Any value.

x0shift Shifts ‘x0‘ away from the center of the category when ‘xref‘ is a "category" or "multicategory" axis. -0.5 corresponds to the start of the category and 0.5 corresponds to the end of the category. Number in the interval [-1, 1].

x1 Sets the shape’s end x position. See ‘type‘ and ‘xsizemode‘ for more info. Any value.

x1shift Shifts ‘x1‘ away from the center of the category when ‘xref‘ is a "category" or "multicategory" axis. -0.5 corresponds to the start of the category and 0.5 corresponds to the end of the category. Number in the interval [-1, 1].

xanchor `go_layout_shape()`: Only relevant in conjunction with ‘xsizemode‘ set to "pixel". Specifies the anchor point on the x axis to which ‘x0‘, ‘x1‘ and x coordinates within ‘path‘ are relative to. E.g. useful to attach a pixel sized shape to a certain data value. No effect when ‘xsizemode‘ not set to "pixel". Any value.

`go_layout_shape_label()`: Sets the label’s horizontal position anchor This anchor binds the specified ‘textposition‘ to the "left", "center" or "right" of the label text. For example, if ‘textposition‘ is set to "top right" and ‘xanchor‘ to "right" then the right-most portion of the label text lines up with the right-most edge of the shape. One of `"auto"`, `"left"`, `"center"`, `"right"`.

xref Sets the shape’s x coordinate axis. If set to a x axis id (e.g. "x" or "x2"), the ‘x‘ position refers to a x coordinate. If set to "paper", the ‘x‘ position refers to the distance from the left of the plotting area in normalized coordinates where 0 (1) corresponds to the left (right). If set to a x axis ID followed by "domain" (separated by a space), the position behaves like for "paper", but refers to the distance in fractions of the domain length from the left of the domain of that axis: e.g., "x2 domain" refers to the domain of the second x axis and a x position of 0.5 refers to the point between the left and the right of the domain of the

second x axis. If an array of axis IDs is provided, each ‘x‘ value will refer to the corresponding axis, e.g., [’x’, ’x2’] for a rectangle, line, or circle means ‘x0‘ uses the ‘x‘ axis and ‘x1‘ uses the ‘x2‘ axis. Path shapes using an array should have one entry for each x coordinate in the string. One of `"paper"`, or a string matching `"^x([2-9]|[1-9][0-9]+)?( domain)?$"`, or a vector of these.

`xsizemode` Sets the shapes’s sizing mode along the x axis. If set to "scaled", ‘x0‘, ‘x1‘ and x coordinates within ‘path‘ refer to data values on the x axis or a fraction of the plot area’s width (‘xref‘ set to "paper"). If set to "pixel", ‘xanchor‘ specifies the x position in terms of data or plot fraction but ‘x0‘, ‘x1‘ and x coordinates within ‘path‘ are pixels relative to ‘xanchor‘. This way, the shape can have a fixed width while maintaining a position relative to data or plot fraction. Note: ‘xsizemode‘ "pixel" is not supported when ‘xref‘ is an array. One of `"scaled"`, `"pixel"`.

`y0` Sets the shape’s starting y position. See ‘type‘ and ‘ysizemode‘ for more info. Any value.

`y0shift` Shifts ‘y0‘ away from the center of the category when ‘yref‘ is a "category" or "multicategory" axis. -0.5 corresponds to the start of the category and 0.5 corresponds to the end of the category. Number in the interval [-1, 1].

`y1` Sets the shape’s end y position. See ‘type‘ and ‘ysizemode‘ for more info. Any value.

`y1shift` Shifts ‘y1‘ away from the center of the category when ‘yref‘ is a "category" or "multicategory" axis. -0.5 corresponds to the start of the category and 0.5 corresponds to the end of the category. Number in the interval [-1, 1].

`yanchor` `go_layout_shape()`: Only relevant in conjunction with ‘ysizemode‘ set to "pixel". Specifies the anchor point on the y axis to which ‘y0‘, ‘y1‘ and y coordinates within ‘path‘ are relative to. E.g. useful to attach a pixel sized shape to a certain data value. No effect when ‘ysizemode‘ not set to "pixel". Any value.

`go_layout_shape_label()`: Sets the label’s vertical position anchor This anchor binds the specified ‘textposition‘ to the "top", "middle" or "bottom" of the label text. For example, if ‘textposition‘ is set to "top right" and ‘yanchor‘ to "top" then the top-most portion of the label text lines up with the top-most edge of the shape. One of `"top"`, `"middle"`, `"bottom"`.

`yref` Sets the shape’s y coordinate axis. If set to a y axis id (e.g. "y" or "y2"), the ‘y‘ position refers to a y coordinate. If set to "paper", the ‘y‘ position refers to the distance from the bottom of the plotting area in normalized coordinates where 0 (1) corresponds to the bottom (top). If set to a y axis ID followed by "domain" (separated by a space), the position behaves like for "paper", but refers to the distance in fractions of the domain length from the bottom of the domain of that axis: e.g., "y2 domain" refers to the domain of the second y axis and a y position of 0.5 refers to the point between the bottom and the top of the domain of the second y axis. If an array of axis IDs is provided, each ‘y‘ value will refer to the corresponding axis, e.g., [’y’, ’y2’] for a rectangle, line, or circle means ‘y0‘ uses the ‘y‘ axis and ‘y1‘ uses the ‘y2‘ axis. Path shapes using an array should have one entry for each y coordinate in the string. One of `"paper"`, or a string matching `"^y([2-9]|[1-9][0-9]+)?( domain)?$"`, or a vector of these.

`ysizemode` Sets the shapes’s sizing mode along the y axis. If set to "scaled", ‘y0‘, ‘y1‘ and y coordinates within ‘path‘ refer to data values on the y axis or a fraction of the

plot area's height ('yref' set to "paper"). If set to "pixel", 'yanchor' specifies the y position in terms of data or plot fraction but 'y0', 'y1' and y coordinates within 'path' are pixels relative to 'yanchor'. This way, the shape can have a fixed height while maintaining a position relative to data or plot fraction. Note: 'ysizemode' "pixel" is not supported when 'yref' is an array. One of `"scaled"`, `"pixel"`.

`row` Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`col` Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`secondary_y` If `TRUE`, use the secondary y axis of the subplot.

`exclude_empty_subplots`
If `TRUE`, skip subplots that have no traces or layout objects.

`skip_invalid` If `TRUE`, silently drop invalid properties instead of raising an error.

`font` `go_layout_shape_label()`: Sets the shape label text font. Named list or `go_layout_shape_label_font()` object.

`go_layout_shape_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_layout_shape_legendgrouptitle_font()` object.

`padding` Sets padding (in px) between edge of label and edge of shape. Number in the interval [0, Inf].

`text` `go_layout_shape_label()`: Sets the text to display with shape. It is also used for legend item if 'name' is not provided. String.

`go_layout_shape_legendgrouptitle()`: Sets the title of the legend group. String.

`textangle` Sets the angle at which the label text is drawn with respect to the horizontal. For lines, angle "auto" is the same angle as the line. For all other shapes, angle "auto" is horizontal. Angle in degrees.

`textposition` Sets the position of the label text relative to the shape. Supported values for rectangles, circles and paths are "top left", "top center", "top right", "middle left", "middle center", "middle right", "bottom left", "bottom center", and "bottom right". Supported values for lines are "start", "middle", and "end". Default: "middle center" for rectangles, circles, and paths; "middle" for lines. One of `"top left"`, `"top center"`, `"top right"`, `"middle left"`, `"middle center"`, `"middle right"`, `"bottom left"`, `"bottom center"`, `"bottom right"`, `"start"`, `"middle"`, `"end"`.

`texttemplate` Template string used for rendering the shape's label. Note that this will override 'text'. Variables are inserted using %{variable}, for example "x0: %{x0}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{x0:$.2f}". See https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{x0|%m %b %Y}". See https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. A single multiplication or division operation may be applied to numeric variables, and combined with d3 number formatting, for example "Length in cm: %{x0"2.54}", "%{slope"60:.1f} meters

per second." For log axes, variable values are given in log units. For date axes, x/y coordinate variables and center variables use datetimes, while all other variable values use values in ms. Finally, the template string has access to variables ‘x0‘, ‘x1‘, ‘y0‘, ‘y1‘, ‘slope‘, ‘dx‘, ‘dy‘, ‘width‘, ‘height‘, ‘length‘, ‘xcenter‘, ‘ycenter‘, ‘simpleXVariables‘ and ‘simpleYVariables‘. Variables that can’t be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. String.

`texttemplatefallback`
: Fallback string that’s displayed when a variable referenced in a template is missing. If the boolean value ’false’ is passed in, the specifier with the missing variable will be displayed. Any value.

`color`
: `go_layout_shape_label_font()`, `go_layout_shape_legendgrouptitle_font()`: Color: a CSS color string or R color name.
`go_layout_shape_line()`: Sets the line color. Color: a CSS color string or R color name.

`family`
: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String.

`lineposition`
: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

`shadow`
: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

`size`
: Number in the interval [1, Inf].

`style`
: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

`textcase`
: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.

`variant`
: Sets the variant of the font. One of `"normal"`, `"small-caps"`, `"all-small-caps"`, `"all-petite-caps"`, `"petite-caps"`, `"unicase"`.

`weight`
: Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

`dash`
: Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

`width`
: Sets the line width (in px). Number in the interval [0, Inf].

Details

`go_layout_shape()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_shape()` creates an object and adds it to a figure.

Nested objects documented here:

`go_layout_shape_label()` `layout.shape.label`

`go_layout_shape_label_font()` `layout.shape.label.font`: Sets the shape label text font.

`go_layout_shape_legendgrouptitle()` `layout.shape.legendgrouptitle`

`go_layout_shape_legendgrouptitle_font()` `layout.shape.legendgrouptitle.font`: Sets this legend group's title font.

`go_layout_shape_line()` `layout.shape.line`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

`add_splom` *Splom traces*

Description

Splom traces generate scatter plot matrix visualizations. Each splom 'dimensions' items correspond to a generated axis. Values for each of those dimensions are set in 'dimensions[i].values'. Splom traces support all 'scattergl' marker style attributes. Specify 'layout.grid' attributes and/or layout x-axis and y-axis attributes for more control over the axis positioning and style.

Usage

```
add_splom(fig, ...)

## Default S3 method:
add_splom(
  fig,
  ...,
  data = NULL,
  customdata = NULL,
  diagonal = NULL,
  dimensions = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
```

```
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  showlowerhalf = NULL,
  showupperhalf = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  xaxes = NULL,
  xhoverformat = NULL,
  yaxes = NULL,
  yhoverformat = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_splom(
  arg = NULL,
  ...,
  customdata = NULL,
  diagonal = NULL,
  dimensions = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
```

```
  opacity = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  showlowerhalf = NULL,
  showupperhalf = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  xaxes = NULL,
  xhoverformat = NULL,
  yaxes = NULL,
  yhoverformat = NULL,
  skip_invalid = FALSE
)

go_splom_diagonal(arg = NULL, ..., visible = NULL, skip_invalid = FALSE)

go_splom_dimension(
  arg = NULL,
  ...,
  axis = NULL,
  label = NULL,
  name = NULL,
  templateitemname = NULL,
  values = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_splom_dimension_axis(
  arg = NULL,
  ...,
  matches = NULL,
  type = NULL,
  skip_invalid = FALSE
)

go_splom_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
```

```
  showarrow = NULL,
  skip_invalid = FALSE
)

go_splom_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_splom_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_splom_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_splom_marker(
  arg = NULL,
  ...,
  angle = NULL,
  autocolorscale = NULL,
  cauto = NULL,
```

```
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  line = NULL,
  opacity = NULL,
  reversescale = NULL,
  showscale = NULL,
  size = NULL,
  sizemin = NULL,
  sizemode = NULL,
  sizeref = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_splom_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
```

```
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_splom_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_splom_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)
```

```
  go_splom_marker_colorbar_title(
    arg = NULL,
    ...,
    font = NULL,
    side = NULL,
    text = NULL,
    skip_invalid = FALSE
  )

  go_splom_marker_colorbar_title_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_splom_marker_line(
    arg = NULL,
    ...,
    autocolorscale = NULL,
    cauto = NULL,
    cmax = NULL,
    cmid = NULL,
    cmin = NULL,
    color = NULL,
    coloraxis = NULL,
    colorscale = NULL,
    reversescale = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_splom_selected(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

  go_splom_selected_marker(
    arg = NULL,
    ...,
    color = NULL,
    opacity = NULL,
```

```
  size = NULL,
  skip_invalid = FALSE
)

go_splom_unselected(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

go_splom_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
diagonal	Named list or go_splom_diagonal() object.
dimensions	List of named lists or go_splom_dimension() objects.
hoverinfo	Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of "x", "y", "z", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_splom_hoverlabel() object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for

details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Same as 'text'. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_splom_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`marker`
: `go_splom()`: Named list or `go_splom_marker()` object.

`go_splom_selected()`: Named list or `go_splom_selected_marker()` object.

`go_splom_unselected()`: Named list or `go_splom_unselected_marker()` object.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use

'%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

name
: go_splom(): Sets the trace name. The trace name appears as the legend item and on hover. String.

go_splom_dimension(), go_splom_marker_colorbar_tickformatstop(): When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

opacity
: go_splom(): Sets the opacity of the trace. Number in the interval [0, 1].

go_splom_marker(): Sets the marker opacity. Number in the interval [0, 1], or a vector of these.

go_splom_selected_marker(): Sets the marker opacity of selected points. Number in the interval [0, 1].

go_splom_unselected_marker(): Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

selected
: Named list or go_splom_selected() object.

selectedpoints
: Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the 'unselected' are turned on for all points, whereas, any other non-array values means no selection all where the 'selected' and 'unselected' styles have no effect. Any value.

showlegend
: Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

showlowerhalf
: Determines whether or not subplots on the lower half from the diagonal are displayed. Logical.

showupperhalf
: Determines whether or not subplots on the upper half from the diagonal are displayed. Logical.

text
: go_splom(): Sets text elements associated with each (x,y) pair to appear on hover. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. String, or a vector of these.

go_splom_legendgrouptitle(): Sets the title of the legend group. String.

go_splom_marker_colorbar_title(): Sets the title of the color bar. String.

uid
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision
: Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can

add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`unselected` Named list or `go_splom_unselected()` object.

`visible` `go_splom()`: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`go_splom_diagonal()`: Determines whether or not subplots on the diagonal are displayed. Logical.

`go_splom_dimension()`: Determines whether or not this dimension is shown on the graph. Note that even visible false dimension contribute to the default grid generate by this splom trace. Logical.

`xaxes` Sets the list of x axes corresponding to dimensions of this splom trace. By default, a splom will match the first N xaxes where N is the number of input dimensions. Note that, in case where ‘diagonal.visible‘ is false and ‘showupperhalf‘ or ‘showlowerhalf‘ is false, this splom trace will generate one less x-axis and one less y-axis. Vector or list of up to 1 element(s).

`xhoverformat` Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

`yaxes` Sets the list of y axes corresponding to dimensions of this splom trace. By default, a splom will match the first N yaxes where N is the number of input dimensions. Note that, in case where ‘diagonal.visible‘ is false and ‘showupperhalf‘ or ‘showlowerhalf‘ is false, this splom trace will generate one less x-axis and one less y-axis. Vector or list of up to 1 element(s).

`yhoverformat` Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

`row` Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`col` Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`skip_invalid` If `TRUE`, silently drop invalid properties instead of raising an error.

`arg` Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

`axis` Named list or `go_splom_dimension_axis()` object.

`label` Sets the label corresponding to this splom dimension. String.

`templateitemname`
Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

`values` Sets the dimension values to be plotted. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`matches` Determines whether or not the x & y axes generated by this dimension match. Equivalent to setting the 'matches' axis attribute in the layout with the correct axis id. Logical.

`type` Sets the axis type for this dimension's generated x and y axes. Note that the axis 'type' values set in layout take precedence over this attribute. One of `"linear"`, `"log"`, `"date"`, `"category"`.

`align` Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`bgcolor` `go_splom_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`go_splom_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

`bordercolor` `go_splom_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`go_splom_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

`font` `go_splom_hoverlabel()`: Sets the font used in hover labels. Named list or `go_splom_hoverlabel_font()` object.

`go_splom_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_splom_legendgrouptitle_font()` object.

`go_splom_marker_colorbar_title()`: Sets this color bar's title font. Named list or `go_splom_marker_colorbar_title_font()` object.

`namelength` Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

`showarrow` Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

`color` `go_splom_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

go_splom_legendgrouptitle_font(), go_splom_marker_colorbar_tickfont(), go_splom_marker_colorbar_title_font(): Color: a CSS color string or R color name.

go_splom_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.cmin‘ and ‘marker.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by splom.marker.colorscale, or a vector of these.

go_splom_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by splom.marker.line.colorscale, or a vector of these.

go_splom_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name.

go_splom_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

family

go_splom_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

go_splom_legendgrouptitle_font(), go_splom_marker_colorbar_tickfont(), go_splom_marker_colorbar_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition

go_splom_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_splom_legendgrouptitle_font(), go_splom_marker_colorbar_tickfont(), go_splom_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow

go_splom_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_splom_legendgrouptitle_font(), go_splom_marker_colorbar_tickfont(), go_splom_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: go_splom_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

  go_splom_legendgrouptitle_font(),go_splom_marker_colorbar_tickfont(), go_splom_marker_colorbar_title_font(): Number in the interval [1, Inf].

  go_splom_marker(): Sets the marker size (in px). Number in the interval [0, Inf], or a vector of these.

  go_splom_selected_marker(): Sets the marker size of selected points. Number in the interval [0, Inf].

  go_splom_unselected_marker(): Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

style
: go_splom_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

  go_splom_legendgrouptitle_font(),go_splom_marker_colorbar_tickfont(), go_splom_marker_colorbar_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_splom_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

  go_splom_legendgrouptitle_font(),go_splom_marker_colorbar_tickfont(), go_splom_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_splom_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

  go_splom_legendgrouptitle_font(),go_splom_marker_colorbar_tickfont(), go_splom_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_splom_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

  go_splom_legendgrouptitle_font(),go_splom_marker_colorbar_tickfont(), go_splom_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

angle
: Sets the marker angle in respect to ‘angleref‘. Angle in degrees, or a vector of these.

autocolorscale
: go_splom_marker(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

  go_splom_marker_line(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘.

Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

`cauto` `go_splom_marker()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

`go_splom_marker_line()`: Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

`cmax` `go_splom_marker()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

`go_splom_marker_line()`: Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

`cmid` `go_splom_marker()`: Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

`go_splom_marker_line()`: Sets the mid-point of the color domain by scaling ‘marker.line.cmin‘ and/or ‘marker.line.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

`cmin` `go_splom_marker()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`go_splom_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_splom_marker_colorbar()` object.

`colorscale` `go_splom_marker()`: Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color

string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the bounds of the colorscale in color space, use 'marker.cmin' and 'marker.cmax'. Alternatively, 'colorscale' may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Re
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_splom_marker_line()`: Sets the colorscale. Has an effect only if in 'marker.line.color' is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the bounds of the colorscale in color space, use 'marker.line.cmin' and 'marker.line.cmax'. Alternatively, 'colorscale' may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`line` Named list or `go_splom_marker_line()` object.

`reversescale` `go_splom_marker()`: Reverses the color mapping if true. Has an effect only if in 'marker.color' is set to a numerical array. If true, 'marker.cmin' will correspond to the last color in the array and 'marker.cmax' will correspond to the first color. Logical.

`go_splom_marker_line()`: Reverses the color mapping if true. Has an effect only if in 'marker.line.color' is set to a numerical array. If true, 'marker.line.cmin' will correspond to the last color in the array and 'marker.line.cmax' will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if in 'marker.color' is set to a numerical array. Logical.

`sizemin` Has an effect only if 'marker.size' is set to a numerical array. Sets the minimum size (in px) of the rendered marker points. Number in the interval [0, Inf].

`sizemode` Has an effect only if 'marker.size' is set to a numerical array. Sets the rule for which the data in 'size' is converted to pixels. One of `"diameter"`, `"area"`.

`sizeref` Has an effect only if 'marker.size' is set to a numerical array. Sets the scale factor used to determine the rendered size of marker points. Use with 'sizemin' and 'sizemode'. Number.

`symbol` Sets the marker symbol type. Adding 100 is equivalent to appending "-open" to a symbol name. Adding 200 is equivalent to appending "-dot" to a symbol name. Adding 300 is equivalent to appending "-open-dot" or "dot-open" to a symbol name. One of `"0"`, `"circle"`, `"100"`, `"circle-open"`, `"200"`, `"circle-dot"`, `"300"`, `"circle-open-dot"`, `"1"`, `"square"`, `"101"`, `"square-open"`, `"201"`, `"square-dot"`, `"301"`, `"square-open-dot"`, `"2"`, `"diamond"`, `"102"`, `"diamond-open"`, `"202"`, `"diamond-dot"`, `"302"`, `"diamond-open-dot"`, `"3"`, `"cross"`, `"103"`, `"cross-open"`, `"203"`, `"cross-dot"`, `"303"`, `"cross-open-dot"`, `"4"`, `"x"`, `"104"`, `"x-open"`, `"204"`, `"x-dot"`, `"304"`, `"x-open-dot"`, `"5"`, `"triangle-up"`,

"105", "triangle-up-open", "205", "triangle-up-dot", "305", "triangle-up-open-dot",
"6", "triangle-down", "106", "triangle-down-open", "206", "triangle-down-dot",
"306", "triangle-down-open-dot", "7", "triangle-left", "107", "triangle-left-open",
"207", "triangle-left-dot", "307", "triangle-left-open-dot", "8", "triangle-right",
"108", "triangle-right-open", "208", "triangle-right-dot", "308", "triangle-right-open-do
"9", "triangle-ne", "109", "triangle-ne-open", "209", "triangle-ne-dot",
"309", "triangle-ne-open-dot", "10", "triangle-se", "110", "triangle-se-open",
"210", "triangle-se-dot", "310", "triangle-se-open-dot", "11", "triangle-sw",
"111", "triangle-sw-open", "211", "triangle-sw-dot", "311", "triangle-sw-open-dot",
"12", "triangle-nw", "112", "triangle-nw-open", "212", "triangle-nw-dot",
"312", "triangle-nw-open-dot", "13", "pentagon", "113", "pentagon-open",
"213", "pentagon-dot", "313", "pentagon-open-dot", "14", "hexagon",
"114", "hexagon-open", "214", "hexagon-dot", "314", "hexagon-open-dot",
"15", "hexagon2", "115", "hexagon2-open", "215", "hexagon2-dot", "315",
"hexagon2-open-dot", "16", "octagon", "116", "octagon-open", "216",
"octagon-dot", "316", "octagon-open-dot", "17", "star", "117", "star-open",
"217", "star-dot", "317", "star-open-dot", "18", "hexagram", "118", "hexagram-open",
"218", "hexagram-dot", "318", "hexagram-open-dot", "19", "star-triangle-up",
"119", "star-triangle-up-open", "219", "star-triangle-up-dot", "319",
"star-triangle-up-open-dot", "20", "star-triangle-down", "120", "star-triangle-down-open
"220", "star-triangle-down-dot", "320", "star-triangle-down-open-dot",
"21", "star-square", "121", "star-square-open", "221", "star-square-dot",
"321", "star-square-open-dot", "22", "star-diamond", "122", "star-diamond-open",
"222", "star-diamond-dot", "322", "star-diamond-open-dot", "23", "diamond-tall",
"123", "diamond-tall-open", "223", "diamond-tall-dot", "323", "diamond-tall-open-dot",
"24", "diamond-wide", "124", "diamond-wide-open", "224", "diamond-wide-dot",
"324", "diamond-wide-open-dot", "25", "hourglass", "125", "hourglass-open",
"26", "bowtie", "126", "bowtie-open", "27", "circle-cross", "127", "circle-cross-open",
"28", "circle-x", "128", "circle-x-open", "29", "square-cross", "129",
"square-cross-open", "30", "square-x", "130", "square-x-open", "31",
"diamond-cross", "131", "diamond-cross-open", "32", "diamond-x", "132",
"diamond-x-open", "33", "cross-thin", "133", "cross-thin-open", "34",
"x-thin", "134", "x-thin-open", "35", "asterisk", "135", "asterisk-open",
"36", "hash", "136", "hash-open", "236", "hash-dot", "336", "hash-open-dot",
"37", "y-up", "137", "y-up-open", "38", "y-down", "138", "y-down-open",
"39", "y-left", "139", "y-left-open", "40", "y-right", "140", "y-right-open",
"41", "line-ew", "141", "line-ew-open", "42", "line-ns", "142", "line-ns-open",
"43", "line-ne", "143", "line-ne-open", "44", "line-nw", "144", "line-nw-open",
"45", "arrow-up", "145", "arrow-up-open", "46", "arrow-down", "146",
"arrow-down-open", "47", "arrow-left", "147", "arrow-left-open", "48",
"arrow-right", "148", "arrow-right-open", "49", "arrow-bar-up", "149",
"arrow-bar-up-open", "50", "arrow-bar-down", "150", "arrow-bar-down-open",
"51", "arrow-bar-left", "151", "arrow-bar-left-open", "52", "arrow-bar-right",
"152", "arrow-bar-right-open", "53", "arrow", "153", "arrow-open", "54",
"arrow-wide", "154", "arrow-wide-open", 0, 100, 200, 300, 1, 101, 201, 301,
2, 102, 202, 302, 3, 103, 203, 303, 4, 104, 204, 304, 5, 105, 205, 305, 6, 106,
206, 306, 7, 107, 207, 307, 8, 108, 208, 308, 9, 109, 209, 309, 10, 110, 210, 310,
11, 111, 211, 311, 12, 112, 212, 312, 13, 113, 213, 313, 14, 114, 214, 314, 15,

115, 215, 315, 16, 116, 216, 316, 17, 117, 217, 317, 18, 118, 218, 318, 19, 119, 219, 319, 20, 120, 220, 320, 21, 121, 221, 321, 22, 122, 222, 322, 23, 123, 223, 323, 24, 124, 224, 324, 25, 125, 26, 126, 27, 127, 28, 128, 29, 129, 30, 130, 31, 131, 32, 132, 33, 133, 34, 134, 35, 135, 36, 136, 236, 336, 37, 137, 38, 138, 39, 139, 40, 140, 41, 141, 42, 142, 43, 143, 44, 144, 45, 145, 46, 146, 47, 147, 48, 148, 49, 149, 50, 150, 51, 151, 52, 152, 53, 153, 54, 154, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick

is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".

showticksuffix Same as 'showtickprefix' but for tick suffixes. One of "all", "first", "last", "none".

thickness Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of "fraction", "pixels".

tick0 Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar's tick label font Named list or go_splom_marker_colorbar_tickfont() object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or go_splom_marker_colorbar_tickformatstop() objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_splom_marker_colorbar_title()` object.

`x` Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`xanchor` Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`y` Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

yanchor	Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of "top", "middle", "bottom".
ypad	Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].
yref	Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".
dtickrange	range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).
enabled	Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.
value	string - dtickformat for described zoom level, the same as "tickformat" String.
side	Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of "right", "top", "bottom".
width	Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

Details

`go_splom()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_splom()` creates an object and adds it to a figure.

Nested objects documented here:

`go_splom_diagonal()` `splom.diagonal`

`go_splom_dimension()` `splom.dimension`

`go_splom_dimension_axis()` `splom.dimension.axis`

`go_splom_hoverlabel()` `splom.hoverlabel`

`go_splom_hoverlabel_font()` `splom.hoverlabel.font`: Sets the font used in hover labels.

`go_splom_legendgrouptitle()` `splom.legendgrouptitle`

`go_splom_legendgrouptitle_font()` `splom.legendgrouptitle.font`: Sets this legend group's title font.

`go_splom_marker()` `splom.marker`

`go_splom_marker_colorbar()` `splom.marker.colorbar`

`go_splom_marker_colorbar_tickfont()` `splom.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_splom_marker_colorbar_tickformatstop()` `splom.marker.colorbar.tickformatstop`

`go_splom_marker_colorbar_title()` `splom.marker.colorbar.title`

`go_splom_marker_colorbar_title_font()` `splom.marker.colorbar.title.font`: Sets this color bar's title font.

`go_splom_marker_line()` `splom.marker.line`

`go_splom_selected()` `splom.selected`

`go_splom_selected_marker()` `splom.selected.marker`

`go_splom_unselected()` `splom.unselected`

`go_splom_unselected_marker()` `splom.unselected.marker`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_splom(marker_color = "red")
```

`add_streamtube` *Streamtube traces*

Description

Use a streamtube trace to visualize flow in a vector field. Specify a vector field using 6 1D arrays of equal length, 3 position arrays 'x', 'y' and 'z' and 3 vector component arrays 'u', 'v', and 'w'. By default, the tubes' starting positions will be cut from the vector field's x-z plane at its minimum y value. To specify your own starting position, use attributes 'starts.x', 'starts.y' and 'starts.z'. The color is encoded by the norm of (u, v, w), and the local radius by the divergence of (u, v, w).

Usage

```
add_streamtube(fig, ...)

## Default S3 method:
add_streamtube(
  fig,
  ...,
  data = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
```

```
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  customdata = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lighting = NULL,
  lightposition = NULL,
  maxdisplayed = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  reversescale = NULL,
  scene = NULL,
  showlegend = NULL,
  showscale = NULL,
  sizeref = NULL,
  starts = NULL,
  text = NULL,
  u = NULL,
  uhoverformat = NULL,
  uid = NULL,
  uirevision = NULL,
  v = NULL,
  vhoverformat = NULL,
  visible = NULL,
  w = NULL,
  whoverformat = NULL,
  x = NULL,
  xhoverformat = NULL,
  y = NULL,
  yhoverformat = NULL,
  z = NULL,
  zhoverformat = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)
```

```
    go_streamtube(
      arg = NULL,
      ...,
      autocolorscale = NULL,
      cauto = NULL,
      cmax = NULL,
      cmid = NULL,
      cmin = NULL,
      coloraxis = NULL,
      colorbar = NULL,
      colorscale = NULL,
      customdata = NULL,
      hoverinfo = NULL,
      hoverlabel = NULL,
      hovertemplate = NULL,
      hovertemplatefallback = NULL,
      hovertext = NULL,
      ids = NULL,
      legend = NULL,
      legendgroup = NULL,
      legendgrouptitle = NULL,
      legendrank = NULL,
      legendwidth = NULL,
      lighting = NULL,
      lightposition = NULL,
      maxdisplayed = NULL,
      meta = NULL,
      name = NULL,
      opacity = NULL,
      reversescale = NULL,
      scene = NULL,
      showlegend = NULL,
      showscale = NULL,
      sizeref = NULL,
      starts = NULL,
      text = NULL,
      u = NULL,
      uhoverformat = NULL,
      uid = NULL,
      uirevision = NULL,
      v = NULL,
      vhoverformat = NULL,
      visible = NULL,
      w = NULL,
      whoverformat = NULL,
      x = NULL,
      xhoverformat = NULL,
      y = NULL,
```

```
    yhoverformat = NULL,
    z = NULL,
    zhoverformat = NULL,
    skip_invalid = FALSE
  )

  go_streamtube_colorbar(
    arg = NULL,
    ...,
    bgcolor = NULL,
    bordercolor = NULL,
    borderwidth = NULL,
    dtick = NULL,
    exponentformat = NULL,
    labelalias = NULL,
    len = NULL,
    lenmode = NULL,
    minexponent = NULL,
    nticks = NULL,
    orientation = NULL,
    outlinecolor = NULL,
    outlinewidth = NULL,
    separatethousands = NULL,
    showexponent = NULL,
    showticklabels = NULL,
    showtickprefix = NULL,
    showticksuffix = NULL,
    thickness = NULL,
    thicknessmode = NULL,
    tick0 = NULL,
    tickangle = NULL,
    tickcolor = NULL,
    tickfont = NULL,
    tickformat = NULL,
    tickformatstops = NULL,
    ticklabeloverflow = NULL,
    ticklabelposition = NULL,
    ticklabelstep = NULL,
    ticklen = NULL,
    tickmode = NULL,
    tickprefix = NULL,
    ticks = NULL,
    ticksuffix = NULL,
    ticktext = NULL,
    tickvals = NULL,
    tickwidth = NULL,
    title = NULL,
    x = NULL,
```

```
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_streamtube_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_streamtube_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_streamtube_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_streamtube_colorbar_title_font(
  arg = NULL,
  ...,
```

```
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_streamtube_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_streamtube_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_streamtube_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_streamtube_legendgrouptitle_font(
```

```
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_streamtube_lighting(
  arg = NULL,
  ...,
  ambient = NULL,
  diffuse = NULL,
  facenormalsepsilon = NULL,
  fresnel = NULL,
  roughness = NULL,
  specular = NULL,
  vertexnormalsepsilon = NULL,
  skip_invalid = FALSE
)

go_streamtube_lightposition(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_streamtube_starts(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.

...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
data	A data frame in which formula properties (e.g. `x = ~col`, `marker_size = ~col`) are evaluated. Default: the data given to `go_figure()`. Only the evaluated values are added to the trace. A `crosstalk::SharedData` works the same way, and links the trace to its crosstalk group (see `highlight()`).
autocolorscale	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
cauto	Determines whether or not the color domain is computed with respect to the input data (here u/v/w norm) or the bounds set in ‘cmin‘ and ‘cmax‘ Defaults to ‘false‘ when ‘cmin‘ and ‘cmax‘ are set by the user. Logical.
cmax	Sets the upper bound of the color domain. Value should have the same units as u/v/w norm and if set, ‘cmin‘ must be set as well. Number.
cmid	Sets the mid-point of the color domain by scaling ‘cmin‘ and/or ‘cmax‘ to be equidistant to this point. Value should have the same units as u/v/w norm. Has no effect when ‘cauto‘ is ‘false‘. Number.
cmin	Sets the lower bound of the color domain. Value should have the same units as u/v/w norm and if set, ‘cmax‘ must be set as well. Number.
coloraxis	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...
colorbar	Named list or `go_streamtube_colorbar()` object.
colorscale	Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘cmin‘ and ‘cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
hoverinfo	Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"u"`, `"v"`, `"w"`, `"norm"`, `"divergence"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_streamtube_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'tubex', 'tubey', 'tubez', 'tubeu', 'tubev', 'tubew', 'norm' and 'divergence'. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Same as 'text'. String.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle` Named list or `go_streamtube_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`lighting` Named list or `go_streamtube_lighting()` object.

`lightposition` Named list or `go_streamtube_lightposition()` object.

`maxdisplayed` The maximum number of displayed segments in a streamtube. Integer in the interval [0, Inf].

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_streamtube()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_streamtube_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` Sets the opacity of the surface. Please note that in the case of using high ‘opacity‘ values for example a value greater than or equal to 0.5 on two surfaces (and 0.25 with four surfaces), an overlay of multiple transparent surfaces may not perfectly be sorted in depth by the webgl API. This behavior may be improved in the near future and is subject to change. Number in the interval [0, 1].

`reversescale` Reverses the color mapping if true. If true, ‘cmin‘ will correspond to the last color in the array and ‘cmax‘ will correspond to the first color. Logical.

`scene` Sets a reference between this trace’s 3D coordinate system and a 3D scene. If "scene" (the default value), the (x,y,z) coordinates refer to ‘layout.scene‘. If "scene2", the (x,y,z) coordinates refer to ‘layout.scene2‘, and so on. Subplot id: `"scene"`, `"scene2"`, `"scene3"`, ...

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Logical.

`sizeref` The scaling factor for the streamtubes. The default is 1, which avoids two max divergence tubes from touching at adjacent starting positions. Number in the interval [0, Inf].

`starts` Named list or `go_streamtube_starts()` object.

`text` `go_streamtube()`: Sets a text element associated with this trace. If trace ‘hoverinfo‘ contains a "text" flag, this text element will be seen in all hover labels. Note that streamtube traces do not support array ‘text‘ values. String.

`go_streamtube_colorbar_title()`: Sets the title of the color bar. String.

`go_streamtube_legendgrouptitle()`: Sets the title of the legend group. String.

u — Sets the x components of the vector field. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

uhoverformat — Sets the hover text formatting rule for ‘u‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

uid — Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision — Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

v — Sets the y components of the vector field. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

vhoverformat — Sets the hover text formatting rule for ‘v‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

visible — Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

w — Sets the z components of the vector field. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

whoverformat — Sets the hover text formatting rule for ‘w‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

x — `go_streamtube()`: Sets the x coordinates of the vector field. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_streamtube_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`go_streamtube_lightposition()`: Numeric vector, representing the X coordinate for each vertex. Number in the interval [-100000, 100000].

`go_streamtube_starts()`: Sets the x components of the starting position of the streamtubes Data array: a vector, factor, Date/POSIXct vector, matrix or list.

xhoverformat Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

y `go_streamtube()`: Sets the y coordinates of the vector field. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_streamtube_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

`go_streamtube_lightposition()`: Numeric vector, representing the Y coordinate for each vertex. Number in the interval [-100000, 100000].

`go_streamtube_starts()`: Sets the y components of the starting position of the streamtubes Data array: a vector, factor, Date/POSIXct vector, matrix or list.

yhoverformat Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

z `go_streamtube()`: Sets the z coordinates of the vector field. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_streamtube_lightposition()`: Numeric vector, representing the Z coordinate for each vertex. Number in the interval [-100000, 100000].

`go_streamtube_starts()`: Sets the z components of the starting position of the streamtubes Data array: a vector, factor, Date/POSIXct vector, matrix or list.

zhoverformat Sets the hover text formatting rule for ‘z‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘zaxis.hoverformat‘. String.

row Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

col Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

skip_invalid — If TRUE, silently drop invalid properties instead of raising an error.

arg — Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

bgcolor — go_streamtube_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name.

go_streamtube_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

bordercolor — go_streamtube_colorbar(): Sets the axis line color. Color: a CSS color string or R color name.

go_streamtube_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

borderwidth — Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

dtick — Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

exponentformat — Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

labelalias — Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

len — Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

lenmode — Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of "fraction", "pixels".

minexponent — Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

nticks — Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

orientation — Sets the orientation of the colorbar. One of "h", "v".

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar’s tick label font Named list or `go_streamtube_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops` List of named lists or `go_streamtube_colorbar_tickformatstop()` objects.

`ticklabeloverflow` Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition`
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep`
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen`
: Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode`
: Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix`
: Sets a tick label prefix. String.

`ticks`
: Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix`
: Sets a tick label suffix. String.

`ticktext`
: Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals`
: Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth`
: Sets the tick width (in px). Number in the interval [0, Inf].

`title`
: String (the title text), named list or `go_streamtube_colorbar_title()` object.

`xanchor`
: Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad`
: Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref`
: Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`yanchor`
: Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

color — `go_streamtube_colorbar_tickfont()`, `go_streamtube_colorbar_title_font()`, `go_streamtube_legendgrouptitle_font()`: Color: a CSS color string or R color name.

`go_streamtube_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

family — `go_streamtube_colorbar_tickfont()`, `go_streamtube_colorbar_title_font()`, `go_streamtube_legendgrouptitle_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String.

`go_streamtube_hoverlabel_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String, or a vector of these.

lineposition — `go_streamtube_colorbar_tickfont()`, `go_streamtube_colorbar_title_font()`, `go_streamtube_legendgrouptitle_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

`go_streamtube_hoverlabel_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

shadow — `go_streamtube_colorbar_tickfont()`, `go_streamtube_colorbar_title_font()`, `go_streamtube_legendgrouptitle_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

`go_streamtube_hoverlabel_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size — `go_streamtube_colorbar_tickfont()`, `go_streamtube_colorbar_title_font()`, `go_streamtube_legendgrouptitle_font()`: Number in the interval [1, Inf].

`go_streamtube_hoverlabel_font()`: Number in the interval [1, Inf], or a vector of these.

style — `go_streamtube_colorbar_tickfont()`, `go_streamtube_colorbar_title_font()`, `go_streamtube_legendgrouptitle_font()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

go_streamtube_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

textcase
go_streamtube_colorbar_tickfont(), go_streamtube_colorbar_title_font(), go_streamtube_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

go_streamtube_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant
go_streamtube_colorbar_tickfont(), go_streamtube_colorbar_title_font(), go_streamtube_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

go_streamtube_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight
go_streamtube_colorbar_tickfont(), go_streamtube_colorbar_title_font(), go_streamtube_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

go_streamtube_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange
range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled
Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname
Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value
string - dtickformat for described zoom level, the same as "tickformat" String.

font
go_streamtube_colorbar_title(): Sets this color bar's title font. Named list or go_streamtube_colorbar_title_font() object.

go_streamtube_hoverlabel(): Sets the font used in hover labels. Named list or go_streamtube_hoverlabel_font() object.

go_streamtube_legendgrouptitle(): Sets this legend group's title font. Named list or go_streamtube_legendgrouptitle_font() object.

side
Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of "right", "top", "bottom".

`align` Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`namelength` Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to ‘namelength - 3‘ characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

`showarrow` Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

`ambient` Ambient light increases overall color visibility but can wash out the image. Number in the interval [0, 1].

`diffuse` Represents the extent that incident rays are reflected in a range of angles. Number in the interval [0, 1].

`facenormalsepsilon` Epsilon for face normals calculation avoids math issues arising from degenerate geometry. Number in the interval [0, 1].

`fresnel` Represents the reflectance as a dependency of the viewing angle; e.g. paper is reflective when viewing it from the edge of the paper (almost 90 degrees), causing shine. Number in the interval [0, 5].

`roughness` Alters specular reflection; the rougher the surface, the wider and less contrasty the shine. Number in the interval [0, 1].

`specular` Represents the level that incident rays are reflected in a single direction, causing shine. Number in the interval [0, 2].

`vertexnormalsepsilon` Epsilon for vertex normals calculation avoids math issues arising from degenerate geometry. Number in the interval [0, 1].

Details

`go_streamtube()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_streamtube()` creates an object and adds it to a figure.

Nested objects documented here:

`go_streamtube_colorbar()` `streamtube.colorbar`

`go_streamtube_colorbar_tickfont()` `streamtube.colorbar.tickfont`: Sets the color bar's tick label font

`go_streamtube_colorbar_tickformatstop()` `streamtube.colorbar.tickformatstop`

`go_streamtube_colorbar_title()` `streamtube.colorbar.title`

`go_streamtube_colorbar_title_font()` `streamtube.colorbar.title.font`: Sets this color bar's title font.

`go_streamtube_hoverlabel()` `streamtube.hoverlabel`

`go_streamtube_hoverlabel_font()` `streamtube.hoverlabel.font`: Sets the font used in hover labels.

`go_streamtube_legendgrouptitle()` `streamtube.legendgrouptitle`

`go_streamtube_legendgrouptitle_font()` `streamtube.legendgrouptitle.font`: Sets this legend group's title font.

`go_streamtube_lighting()` `streamtube.lighting`

`go_streamtube_lightposition()` `streamtube.lightposition`

`go_streamtube_starts()` `streamtube.starts`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_streamtube(x = 1:3, y = c(2, 1, 3))
```

---

add_sunburst *Sunburst traces*

Description

Visualize hierarchal data spanning outward radially from root to leaves. The sunburst sectors are determined by the entries in "labels" or "ids" and in "parents".

Usage

```
add_sunburst(fig, ...)

## Default S3 method:
add_sunburst(
  fig,
  ...,
  data = NULL,
  branchvalues = NULL,
  count = NULL,
  customdata = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
```

```
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextfont = NULL,
  insidetextorientation = NULL,
  labels = NULL,
  leaf = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  level = NULL,
  marker = NULL,
  maxdepth = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  outsidetextfont = NULL,
  parents = NULL,
  root = NULL,
  rotation = NULL,
  sort = NULL,
  text = NULL,
  textfont = NULL,
  textinfo = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  values = NULL,
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_sunburst(
  arg = NULL,
  ...,
  branchvalues = NULL,
  count = NULL,
  customdata = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
```

```
  ids = NULL,
  insidetextfont = NULL,
  insidetextorientation = NULL,
  labels = NULL,
  leaf = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  level = NULL,
  marker = NULL,
  maxdepth = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  outsidetextfont = NULL,
  parents = NULL,
  root = NULL,
  rotation = NULL,
  sort = NULL,
  text = NULL,
  textfont = NULL,
  textinfo = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  values = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_sunburst_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_sunburst_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
```

```
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_sunburst_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_sunburst_insidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_sunburst_leaf(arg = NULL, ..., opacity = NULL, skip_invalid = FALSE)

go_sunburst_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_sunburst_legendgrouptitle_font(
  arg = NULL,
```

```
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_sunburst_marker(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colors = NULL,
  colorscale = NULL,
  line = NULL,
  pattern = NULL,
  reversescale = NULL,
  showscale = NULL,
  skip_invalid = FALSE
)

go_sunburst_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
```

```
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_sunburst_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
```

```
  skip_invalid = FALSE
)

go_sunburst_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_sunburst_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_sunburst_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_sunburst_marker_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_sunburst_marker_pattern(
  arg = NULL,
```

```
  ...,
  bgcolor = NULL,
  fgcolor = NULL,
  fgopacity = NULL,
  fillmode = NULL,
  path = NULL,
  shape = NULL,
  size = NULL,
  solidity = NULL,
  skip_invalid = FALSE
)

go_sunburst_outsidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_sunburst_root(arg = NULL, ..., color = NULL, skip_invalid = FALSE)

go_sunburst_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g.

marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.

data — A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).

branchvalues — Determines how the items in ‘values‘ are summed. When set to "total", items in ‘values‘ are taken to be value of all its descendants. When set to "remainder", items in ‘values‘ corresponding to the root and the branches sectors are taken to be the extra part not part of the sum of the values at their leaves. One of "remainder", "total".

count — Determines default for ‘values‘ when it is not provided, by inferring a 1 for each of the "leaves" and/or "branches", otherwise 0. Any combination of "branches", "leaves" joined with "+".

customdata — Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

domain — Named list or go_sunburst_domain() object.

hoverinfo — Determines what trace information appears on hover. Flags are rendered in a fixed order; use ‘hovertemplate‘ if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of "label", "text", "value", "name", "current path", "percent root", "percent entry", "percent parent" joined with "+", or one of "all", "none", "skip", or a vector of these.

hoverlabel — Named list or go_sunburst_hoverlabel() object.

hovertemplate — Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variables ‘currentPath‘, ‘root‘, ‘entry‘, ‘percentRoot‘, ‘percentEntry‘ and ‘percentParent‘. Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example

‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that’s displayed when a variable referenced in a template is missing. If the boolean value ’false’ is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each sector. If a single string, the same string appears for all data points. If an array of string, the items are mapped in order of this trace’s sectors. To be seen, trace ‘hoverinfo‘ must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`insidetextfont`
: Sets the font used for ‘textinfo‘ lying inside the sector. Named list or `go_sunburst_insidetextfont()` object.

`insidetextorientation`
: Controls the orientation of the text inside chart sectors. When set to "auto", text may be oriented in any direction in order to be as big as possible in the middle of a sector. The "horizontal" option orients text to be parallel with the bottom of the chart, and may make text smaller in order to achieve that goal. The "radial" option orients text along the radius of the sector. The "tangential" option orients text perpendicular to the radius of the sector. One of `"horizontal"`, `"radial"`, `"tangential"`, `"auto"`.

`labels`
: Sets the labels of each of the sectors. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`leaf`
: Named list or `go_sunburst_leaf()` object.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgrouptitle`
: Named list or `go_sunburst_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`level`
: Sets the level from which this trace hierarchy is rendered. Set ‘level‘ to ‘”‘ to start from the root node in the hierarchy. Must be an "id" if ‘ids‘ is filled in, otherwise plotly attempts to find a matching item in ‘labels‘. Any value.

`marker`
: Named list or `go_sunburst_marker()` object.

`maxdepth` Sets the number of rendered sectors from any given ‘level‘. Set ‘maxdepth‘ to "-1" to render all the levels in the hierarchy. Integer.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_sunburst()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_sunburst_marker_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` `go_sunburst()`: Sets the opacity of the trace. Number in the interval [0, 1].

`go_sunburst_leaf()`: Sets the opacity of the leaves. With colorscale it is defaulted to 1; otherwise it is defaulted to 0.7 Number in the interval [0, 1].

`outsidetextfont`

Sets the font used for ‘textinfo‘ lying outside the sector. This option refers to the root of the hierarchy presented at the center of a sunburst graph. Please note that if a hierarchy has multiple root nodes, this option won't have any effect and ‘insidetextfont‘ would be used. Named list or `go_sunburst_outsidetextfont()` object.

`parents` Sets the parent sectors for each of the sectors. Empty string items ” are understood to reference the root node in the hierarchy. If ‘ids‘ is filled, ‘parents‘ items are understood to be "ids" themselves. When ‘ids‘ is not set, plotly attempts to find matching items in ‘labels‘, but beware they must be unique. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`root` Named list or `go_sunburst_root()` object.

`rotation` Rotates the whole diagram counterclockwise by some angle. By default the first slice starts at 3 o'clock. Angle in degrees.

`sort` Determines whether or not the sectors are reordered from largest to smallest. Logical.

`text` `go_sunburst()`: Sets text elements associated with each sector. If trace ‘textinfo‘ contains a "text" flag, these elements will be seen on the chart. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_sunburst_legendgrouptitle()`: Sets the title of the legend group. String.

`go_sunburst_marker_colorbar_title()`: Sets the title of the color bar. String.

`textfont` Sets the font used for ‘textinfo‘. Named list or `go_sunburst_textfont()` object.

`textinfo` Determines what trace information appears on the graph. Flags are rendered in a fixed order; use 'texttemplate' if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of `"label"`, `"text"`, `"value"`, `"current path"`, `"percent root"`, `"percent entry"`, `"percent parent"` joined with `"+"`, or one of `"none"`.

`texttemplate` Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'currentPath', 'root', 'entry', 'percentRoot', 'percentEntry', 'percentParent', 'label' and 'value'. String, or a vector of these.

`texttemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

`values` Sets the values associated with each of the sectors. Use with 'branchvalues' to determine how the values are summed. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`visible` Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`row` `add_sunburst()`: Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`go_sunburst_domain()`: If there is a layout grid, use the domain for this row in the grid for this sunburst trace. Integer in the interval [0, Inf].

col — Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

skip_invalid — If TRUE, silently drop invalid properties instead of raising an error.

arg — Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

column — If there is a layout grid, use the domain for this column in the grid for this sunburst trace. Integer in the interval [0, Inf].

x — go_sunburst_domain(): Sets the horizontal domain of this sunburst trace (in plot fraction). Vector or list of 2 element(s).

go_sunburst_marker_colorbar(): Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

y — go_sunburst_domain(): Sets the vertical domain of this sunburst trace (in plot fraction). Vector or list of 2 element(s).

go_sunburst_marker_colorbar(): Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

align — Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.

bgcolor — go_sunburst_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

go_sunburst_marker_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name.

go_sunburst_marker_pattern(): When there is no colorscale sets the color of background pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "overlay". Otherwise, defaults to a transparent background. Color: a CSS color string or R color name, or a vector of these.

bordercolor — go_sunburst_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

go_sunburst_marker_colorbar(): Sets the axis line color. Color: a CSS color string or R color name.

font — go_sunburst_hoverlabel(): Sets the font used in hover labels. Named list or go_sunburst_hoverlabel_font() object.

go_sunburst_legendgrouptitle(): Sets this legend group’s title font. Named list or go_sunburst_legendgrouptitle_font() object.

go_sunburst_marker_colorbar_title(): Sets this color bar’s title font. Named list or go_sunburst_marker_colorbar_title_font() object.

namelength — Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to ‘namelength - 3‘ characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow — Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color — `go_sunburst_hoverlabel_font()`, `go_sunburst_insidetextfont()`, `go_sunburst_outsidetextfo` `go_sunburst_textfont()`: Color: a CSS color string or R color name, or a vector of these.

`go_sunburst_legendgrouptitle_font()`, `go_sunburst_marker_colorbar_tickfont()`, `go_sunburst_marker_colorbar_title_font()`: Color: a CSS color string or R color name.

`go_sunburst_marker_line()`: Sets the color of the line enclosing each sector. Defaults to the ‘paper_bgcolor‘ value. Color: a CSS color string or R color name, or a vector of these.

`go_sunburst_root()`: sets the color of the root node for a sunburst/treemap/icicle trace. this has no effect when a colorscale is used to set the markers. Color: a CSS color string or R color name.

family — `go_sunburst_hoverlabel_font()`, `go_sunburst_insidetextfont()`, `go_sunburst_outsidetextfo` `go_sunburst_textfont()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String, or a vector of these.

`go_sunburst_legendgrouptitle_font()`, `go_sunburst_marker_colorbar_tickfont()`, `go_sunburst_marker_colorbar_title_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String.

lineposition — `go_sunburst_hoverlabel_font()`, `go_sunburst_insidetextfont()`, `go_sunburst_outsidetextfo` `go_sunburst_textfont()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

`go_sunburst_legendgrouptitle_font()`, `go_sunburst_marker_colorbar_tickfont()`, `go_sunburst_marker_colorbar_title_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

shadow — `go_sunburst_hoverlabel_font()`, `go_sunburst_insidetextfont()`, `go_sunburst_outsidetextfo` `go_sunburst_textfont()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_sunburst_legendgrouptitle_font(), go_sunburst_marker_colorbar_tickfont(), go_sunburst_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
go_sunburst_hoverlabel_font(), go_sunburst_insidetextfont(), go_sunburst_outsidetextfo go_sunburst_textfont(): Number in the interval [1, Inf], or a vector of these.

go_sunburst_legendgrouptitle_font(), go_sunburst_marker_colorbar_tickfont(), go_sunburst_marker_colorbar_title_font(): Number in the interval [1, Inf].

go_sunburst_marker_pattern(): Sets the size of unit squares of the pattern fill in pixels, which corresponds to the interval of repetition of the pattern. Number in the interval [0, Inf], or a vector of these.

style
go_sunburst_hoverlabel_font(), go_sunburst_insidetextfont(), go_sunburst_outsidetextfo go_sunburst_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_sunburst_legendgrouptitle_font(), go_sunburst_marker_colorbar_tickfont(), go_sunburst_marker_colorbar_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
go_sunburst_hoverlabel_font(), go_sunburst_insidetextfont(), go_sunburst_outsidetextfo go_sunburst_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_sunburst_legendgrouptitle_font(), go_sunburst_marker_colorbar_tickfont(), go_sunburst_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
go_sunburst_hoverlabel_font(), go_sunburst_insidetextfont(), go_sunburst_outsidetextfo go_sunburst_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_sunburst_legendgrouptitle_font(), go_sunburst_marker_colorbar_tickfont(), go_sunburst_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
go_sunburst_hoverlabel_font(), go_sunburst_insidetextfont(), go_sunburst_outsidetextfo go_sunburst_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_sunburst_legendgrouptitle_font(), go_sunburst_marker_colorbar_tickfont(), go_sunburst_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

autocolorscale
Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if colors is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

cauto	Determines whether or not the color domain is computed with respect to the input data (here colors) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if colors is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.
cmax	Sets the upper bound of the color domain. Has an effect only if colors is set to a numerical array. Value should have the same units as colors and if set, ‘marker.cmin‘ must be set as well. Number.
cmid	Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if colors is set to a numerical array. Value should have the same units as colors. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.
cmin	Sets the lower bound of the color domain. Has an effect only if colors is set to a numerical array. Value should have the same units as colors and if set, ‘marker.cmax‘ must be set as well. Number.
coloraxis	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...
colorbar	Named list or `go_sunburst_marker_colorbar()` object.
colors	Sets the color of each sector of this trace. If not specified, the default trace color set is used to pick the sector colors. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
colorscale	Sets the colorscale. Has an effect only if colors is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).
line	Named list or `go_sunburst_marker_line()` object.
pattern	Sets the pattern within the marker. Named list or `go_sunburst_marker_pattern()` object.
reversescale	Reverses the color mapping if true. Has an effect only if colors is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.
showscale	Determines whether or not a colorbar is displayed for this trace. Has an effect only if colors is set to a numerical array. Logical.
borderwidth	Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].
dtick	Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string

forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_sunburst_marker_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops` List of named lists or `go_sunburst_marker_colorbar_tickformatstop()` objects.

`ticklabeloverflow` Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition` Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are

provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_sunburst_marker_colorbar_title()` object.

`xanchor` Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`yanchor` Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

`ypad` Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

`yref` Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

`dtickrange` range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

`enabled` Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

`templateitemname`
Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching

	its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.
value	string - dtickformat for described zoom level, the same as "tickformat" String.
side	Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.
width	Sets the width (in px) of the line enclosing each sector. Number in the interval [0, Inf], or a vector of these.
fgcolor	When there is no colorscale sets the color of foreground pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "replace". Otherwise, defaults to dark grey or white to increase contrast with the ‘bgcolor‘. Color: a CSS color string or R color name, or a vector of these.
fgopacity	Sets the opacity of the foreground pattern fill. Defaults to a 0.5 when ‘fillmode‘ is "overlay". Otherwise, defaults to 1. Number in the interval [0, 1].
fillmode	Determines whether ‘marker.color‘ should be used as a default to ‘bgcolor‘ or a ‘fgcolor‘. One of `"replace"`, `"overlay"`.
path	Sets a custom path for pattern fill. Use with no ‘shape‘ or ‘solidity‘, provide an SVG path string for the regions of the square from (0,0) to (‘size‘,‘size‘) to color. String, or a vector of these.
shape	Sets the shape of the pattern fill. By default, no pattern is used for filling the area. One of `""`, `"/"`, `"\\"`, `"x"`, `"-"`, `"|"`, `"+"`, `"."`, or a vector of these.
solidity	Sets the solidity of the pattern fill. Solidity is roughly the fraction of the area filled by the pattern. Solidity of 0 shows only the background color without pattern and solidty of 1 shows only the foreground color without pattern. Number in the interval [0, 1], or a vector of these.

Details

`go_sunburst()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_sunburst()` creates an object and adds it to a figure.

Nested objects documented here:

`go_sunburst_domain()` `sunburst.domain`

`go_sunburst_hoverlabel()` `sunburst.hoverlabel`

`go_sunburst_hoverlabel_font()` `sunburst.hoverlabel.font`: Sets the font used in hover labels.

`go_sunburst_insidetextfont()` `sunburst.insidetextfont`: Sets the font used for ‘textinfo‘ lying inside the sector.

`go_sunburst_leaf()` `sunburst.leaf`

`go_sunburst_legendgrouptitle()` `sunburst.legendgrouptitle`

`go_sunburst_legendgrouptitle_font()` `sunburst.legendgrouptitle.font`: Sets this legend group's title font.

`go_sunburst_marker()` `sunburst.marker`

`go_sunburst_marker_colorbar()` `sunburst.marker.colorbar`

`go_sunburst_marker_colorbar_tickfont()` `sunburst.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_sunburst_marker_colorbar_tickformatstop()` `sunburst.marker.colorbar.tickformatstop`

`go_sunburst_marker_colorbar_title()` `sunburst.marker.colorbar.title`

`go_sunburst_marker_colorbar_title_font()` `sunburst.marker.colorbar.title.font`: Sets this color bar's title font.

`go_sunburst_marker_line()` `sunburst.marker.line`

`go_sunburst_marker_pattern()` `sunburst.marker.pattern`: Sets the pattern within the marker.

`go_sunburst_outsidetextfont()` `sunburst.outsidetextfont`: Sets the font used for 'textinfo' lying outside the sector. This option refers to the root of the hierarchy presented at the center of a sunburst graph. Please note that if a hierarchy has multiple root nodes, this option won't have any effect and 'insidetextfont' would be used.

`go_sunburst_root()` `sunburst.root`

`go_sunburst_textfont()` `sunburst.textfont`: Sets the font used for 'textinfo'.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_sunburst()
```

---

add_surface *Surface traces*

Description

The data the describes the coordinates of the surface is set in 'z'. Data in 'z' should be a matrix. Coordinates in 'x' and 'y' can either be 1D vectors or matrices (e.g. to graph parametric surfaces). If not provided in 'x' and 'y', the x and y coordinates are assumed to be linear starting at 0 with a unit step. The color scale corresponds to the 'z' values by default. For custom color scales, use 'surfacecolor' which should be a matrix, where its bounds can be controlled using 'cmin' and 'cmax'.

Usage

```
add_surface(fig, ...)

## Default S3 method:
add_surface(
  fig,
  ...,
  data = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  connectgaps = NULL,
  contours = NULL,
  customdata = NULL,
  hidesurface = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lighting = NULL,
  lightposition = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  opacityscale = NULL,
  reversescale = NULL,
  scene = NULL,
  showlegend = NULL,
  showscale = NULL,
  surfacecolor = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  xcalendar = NULL,
```

```
  xhoverformat = NULL,
  y = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  z = NULL,
  zcalendar = NULL,
  zhoverformat = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_surface(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  connectgaps = NULL,
  contours = NULL,
  customdata = NULL,
  hidesurface = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lighting = NULL,
  lightposition = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  opacityscale = NULL,
  reversescale = NULL,
  scene = NULL,
  showlegend = NULL,
  showscale = NULL,
```

```
  surfacecolor = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  y = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  z = NULL,
  zcalendar = NULL,
  zhoverformat = NULL,
  skip_invalid = FALSE
)

go_surface_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
```

```
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_surface_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_surface_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_surface_colorbar_title(
```

```
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_surface_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_surface_contours(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_surface_contours_x(
  arg = NULL,
  ...,
  color = NULL,
  end = NULL,
  highlight = NULL,
  highlightcolor = NULL,
  highlightwidth = NULL,
  project = NULL,
  show = NULL,
  size = NULL,
  start = NULL,
  usecolormap = NULL,
  width = NULL,
  skip_invalid = FALSE
)
```

```
  go_surface_contours_x_project(
    arg = NULL,
    ...,
    x = NULL,
    y = NULL,
    z = NULL,
    skip_invalid = FALSE
  )

  go_surface_contours_y(
    arg = NULL,
    ...,
    color = NULL,
    end = NULL,
    highlight = NULL,
    highlightcolor = NULL,
    highlightwidth = NULL,
    project = NULL,
    show = NULL,
    size = NULL,
    start = NULL,
    usecolormap = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_surface_contours_y_project(
    arg = NULL,
    ...,
    x = NULL,
    y = NULL,
    z = NULL,
    skip_invalid = FALSE
  )

  go_surface_contours_z(
    arg = NULL,
    ...,
    color = NULL,
    end = NULL,
    highlight = NULL,
    highlightcolor = NULL,
    highlightwidth = NULL,
    project = NULL,
    show = NULL,
    size = NULL,
    start = NULL,
```

```
  usecolormap = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_surface_contours_z_project(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_surface_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_surface_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_surface_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)
```

```
go_surface_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_surface_lighting(
  arg = NULL,
  ...,
  ambient = NULL,
  diffuse = NULL,
  fresnel = NULL,
  roughness = NULL,
  specular = NULL,
  skip_invalid = FALSE
)

go_surface_lightposition(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)
```

Arguments

`fig`	A `go_figure` object.
`...`	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
`data`	A data frame in which formula properties (e.g. `x = ~col`, `marker_size = ~col`) are evaluated. Default: the data given to `go_figure()`. Only the evaluated values are added to the trace. A `crosstalk::SharedData` works the same way, and links the trace to its crosstalk group (see `highlight()`).
`autocolorscale`	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or

'autocolorscale' is true, the default palette will be chosen according to whether numbers in the 'color' array are all positive, all negative or mixed. Logical.

`cauto` Determines whether or not the color domain is computed with respect to the input data (here z or surfacecolor) or the bounds set in 'cmin' and 'cmax' Defaults to 'false' when 'cmin' and 'cmax' are set by the user. Logical.

`cmax` Sets the upper bound of the color domain. Value should have the same units as z or surfacecolor and if set, 'cmin' must be set as well. Number.

`cmid` Sets the mid-point of the color domain by scaling 'cmin' and/or 'cmax' to be equidistant to this point. Value should have the same units as z or surfacecolor. Has no effect when 'cauto' is 'false'. Number.

`cmin` Sets the lower bound of the color domain. Value should have the same units as z or surfacecolor and if set, 'cmax' must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under 'layout.coloraxis', 'layout.coloraxis2', etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_surface_colorbar()` object.

`colorscale` Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the bounds of the colorscale in color space, use 'cmin' and 'cmax'. Alternatively, 'colorscale' may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`connectgaps` Determines whether or not gaps (i.e. {nan} or missing values) in the 'z' data are filled in. Logical.

`contours` Named list or `go_surface_contours()` object.

`customdata` Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`hidesurface` Determines whether or not a surface is drawn. For example, set 'hidesurface' to FALSE 'contours.x.show' to TRUE and 'contours.y.show' to TRUE to draw a wire frame plot. Logical.

`hoverinfo` Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_surface_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example ‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

`hovertemplatefallback`
Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Same as ‘text‘. String, or a vector of these.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
Named list or `go_surface_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`lighting` Named list or `go_surface_lighting()` object.

`lightposition` Named list or `go_surface_lightposition()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_surface()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_surface_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` Sets the opacity of the surface. Please note that in the case of using high ‘opacity‘ values for example a value greater than or equal to 0.5 on two surfaces (and 0.25 with four surfaces), an overlay of multiple transparent surfaces may not perfectly be sorted in depth by the webgl API. This behavior may be improved in the near future and is subject to change. Number in the interval [0, 1].

`opacityscale` Sets the opacityscale. The opacityscale must be an array containing arrays mapping a normalized value to an opacity value. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, 1], [0.5, 0.2], [1, 1]]‘ means that higher/lower values would have higher opacity values and those in the middle would be more transparent Alternatively, ‘opacityscale‘ may be a palette name string of the following list: ’min’, ’max’, ’extremes’ and ’uniform’. The default is ’uniform’. Any value.

`reversescale` Reverses the color mapping if true. If true, ‘cmin‘ will correspond to the last color in the array and ‘cmax‘ will correspond to the first color. Logical.

`scene` Sets a reference between this trace’s 3D coordinate system and a 3D scene. If "scene" (the default value), the (x,y,z) coordinates refer to ‘layout.scene‘. If "scene2", the (x,y,z) coordinates refer to ‘layout.scene2‘, and so on. Subplot id: `"scene"`, `"scene2"`, `"scene3"`, ...

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Logical.

`surfacecolor` Sets the surface color values, used for setting a color scale independent of ‘z‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`text` `go_surface()`: Sets the text elements associated with each z value. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_surface_colorbar_title()`: Sets the title of the color bar. String.

`go_surface_legendgrouptitle()`: Sets the title of the legend group. String.

uid — Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision — Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

visible — Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

x — `go_surface()`: Sets the x coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_surface_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`go_surface_contours()`: Named list or `go_surface_contours_x()` object.

`go_surface_contours_x_project()`, `go_surface_contours_y_project()`, `go_surface_contours_z_project()`: Determines whether or not these contour lines are projected on the x plane. If ‘highlight‘ is set to TRUE (the default), the projected lines are shown on hover. If ‘show‘ is set to TRUE, the projected lines are shown in permanence. Logical.

`go_surface_lightposition()`: Numeric vector, representing the X coordinate for each vertex. Number in the interval [-100000, 100000].

xcalendar — Sets the calendar system to use with ‘x‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

xhoverformat — Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

y — `go_surface()`: Sets the y coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_surface_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘

is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

`go_surface_contours()`: Named list or `go_surface_contours_y()` object.

`go_surface_contours_x_project()`, `go_surface_contours_y_project()`, `go_surface_contours_z_project()`: Determines whether or not these contour lines are projected on the y plane. If ‘highlight‘ is set to TRUE (the default), the projected lines are shown on hover. If ‘show‘ is set to TRUE, the projected lines are shown in permanence. Logical.

`go_surface_lightposition()`: Numeric vector, representing the Y coordinate for each vertex. Number in the interval [-100000, 100000].

`ycalendar` Sets the calendar system to use with ‘y‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

`yhoverformat` Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

`z` `go_surface()`: Sets the z coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_surface_contours()`: Named list or `go_surface_contours_z()` object.

`go_surface_contours_x_project()`, `go_surface_contours_y_project()`, `go_surface_contours_z_project()`: Determines whether or not these contour lines are projected on the z plane. If ‘highlight‘ is set to TRUE (the default), the projected lines are shown on hover. If ‘show‘ is set to TRUE, the projected lines are shown in permanence. Logical.

`go_surface_lightposition()`: Numeric vector, representing the Z coordinate for each vertex. Number in the interval [-100000, 100000].

`zcalendar` Sets the calendar system to use with ‘z‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

`zhoverformat` Sets the hover text formatting rule for ‘z‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘zaxis.hoverformat‘. String.

`row` Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

col — Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

skip_invalid — If TRUE, silently drop invalid properties instead of raising an error.

arg — Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

bgcolor — go_surface_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name.

go_surface_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

bordercolor — go_surface_colorbar(): Sets the axis line color. Color: a CSS color string or R color name.

go_surface_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

borderwidth — Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

dtick — Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

exponentformat — Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

labelalias — Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

len — Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

lenmode — Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of "fraction", "pixels".

minexponent — Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

nticks — Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands`
If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar’s tick label font Named list or `go_surface_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops`
List of named lists or `go_surface_colorbar_tickformatstop()` objects.

`ticklabeloverflow`
Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past

domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition`
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep`
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

`ticklen`
: Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode`
: Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix`
: Sets a tick label prefix. String.

`ticks`
: Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix`
: Sets a tick label suffix. String.

`ticktext`
: Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals`
: Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth`
: Sets the tick width (in px). Number in the interval [0, Inf].

`title`
: String (the title text), named list or `go_surface_colorbar_title()` object.

`xanchor`
: Sets this color bar's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the color bar. Defaults to "left" when 'orientation' is "v" and "center" when 'orientation' is "h". One of `"left"`, `"center"`, `"right"`.

`xpad`
: Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref`
: Sets the container 'x' refers to. "container" spans the entire 'width' of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

yanchor — Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of `"top"`, `"middle"`, `"bottom"`.

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

color — `go_surface_colorbar_tickfont()`, `go_surface_colorbar_title_font()`, `go_surface_legendgrouptitle_font()`: Color: a CSS color string or R color name.

`go_surface_contours_x()`, `go_surface_contours_y()`, `go_surface_contours_z()`: Sets the color of the contour lines. Color: a CSS color string or R color name.

`go_surface_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

family — `go_surface_colorbar_tickfont()`, `go_surface_colorbar_title_font()`, `go_surface_legendgrouptitle_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

`go_surface_hoverlabel_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

lineposition — `go_surface_colorbar_tickfont()`, `go_surface_colorbar_title_font()`, `go_surface_legendgrouptitle_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

`go_surface_hoverlabel_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

shadow — `go_surface_colorbar_tickfont()`, `go_surface_colorbar_title_font()`, `go_surface_legendgrouptitle_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

`go_surface_hoverlabel_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size
: go_surface_colorbar_tickfont(), go_surface_colorbar_title_font(), go_surface_legendgrouptitle_font(): Number in the interval [1, Inf].

  go_surface_contours_x(), go_surface_contours_y(), go_surface_contours_z(): Sets the step between each contour level. Must be positive. Number in the interval [0, Inf].

  go_surface_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

style
: go_surface_colorbar_tickfont(), go_surface_colorbar_title_font(), go_surface_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

  go_surface_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

textcase
: go_surface_colorbar_tickfont(), go_surface_colorbar_title_font(), go_surface_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

  go_surface_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant
: go_surface_colorbar_tickfont(), go_surface_colorbar_title_font(), go_surface_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

  go_surface_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight
: go_surface_colorbar_tickfont(), go_surface_colorbar_title_font(), go_surface_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

  go_surface_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange
: range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled
: Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname
: Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value
: string - dtickformat for described zoom level, the same as "tickformat" String.

font	go_surface_colorbar_title(): Sets this color bar's title font. Named list or go_surface_colorbar_title_font() object. go_surface_hoverlabel(): Sets the font used in hover labels. Named list or go_surface_hoverlabel_font() object. go_surface_legendgrouptitle(): Sets this legend group's title font. Named list or go_surface_legendgrouptitle_font() object.
side	Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of "right", "top", "bottom".
end	Sets the end contour level value. Must be more than 'contours.start' Number.
highlight	go_surface_contours_x(): Determines whether or not contour lines about the x dimension are highlighted on hover. Logical. go_surface_contours_y(): Determines whether or not contour lines about the y dimension are highlighted on hover. Logical. go_surface_contours_z(): Determines whether or not contour lines about the z dimension are highlighted on hover. Logical.
highlightcolor	Sets the color of the highlighted contour lines. Color: a CSS color string or R color name.
highlightwidth	Sets the width of the highlighted contour lines. Number in the interval [1, 16].
project	go_surface_contours_x(): Named list or go_surface_contours_x_project() object. go_surface_contours_y(): Named list or go_surface_contours_y_project() object. go_surface_contours_z(): Named list or go_surface_contours_z_project() object.
show	go_surface_contours_x(): Determines whether or not contour lines about the x dimension are drawn. Logical. go_surface_contours_y(): Determines whether or not contour lines about the y dimension are drawn. Logical. go_surface_contours_z(): Determines whether or not contour lines about the z dimension are drawn. Logical.
start	Sets the starting contour level value. Must be less than 'contours.end' Number.
usecolormap	An alternate to "color". Determines whether or not the contour lines are colored using the trace "colorscale". Logical.
width	Sets the width of the contour lines. Number in the interval [1, 16].
align	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.
namelength	Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow	Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.
ambient	Ambient light increases overall color visibility but can wash out the image. Number in the interval [0, 1].
diffuse	Represents the extent that incident rays are reflected in a range of angles. Number in the interval [0, 1].
fresnel	Represents the reflectance as a dependency of the viewing angle; e.g. paper is reflective when viewing it from the edge of the paper (almost 90 degrees), causing shine. Number in the interval [0, 5].
roughness	Alters specular reflection; the rougher the surface, the wider and less contrasty the shine. Number in the interval [0, 1].
specular	Represents the level that incident rays are reflected in a single direction, causing shine. Number in the interval [0, 2].

Details

`go_surface()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_surface()` creates an object and adds it to a figure.

Nested objects documented here:

`go_surface_colorbar()` `surface.colorbar`

`go_surface_colorbar_tickfont()` `surface.colorbar.tickfont`: Sets the color bar's tick label font

`go_surface_colorbar_tickformatstop()` `surface.colorbar.tickformatstop`

`go_surface_colorbar_title()` `surface.colorbar.title`

`go_surface_colorbar_title_font()` `surface.colorbar.title.font`: Sets this color bar's title font.

`go_surface_contours()` `surface.contours`

`go_surface_contours_x()` `surface.contours.x`

`go_surface_contours_x_project()` `surface.contours.x.project`

`go_surface_contours_y()` `surface.contours.y`

`go_surface_contours_y_project()` `surface.contours.y.project`

`go_surface_contours_z()` `surface.contours.z`

`go_surface_contours_z_project()` `surface.contours.z.project`

`go_surface_hoverlabel()` `surface.hoverlabel`

`go_surface_hoverlabel_font()` `surface.hoverlabel.font`: Sets the font used in hover labels.

`go_surface_legendgrouptitle()` `surface.legendgrouptitle`

`go_surface_legendgrouptitle_font()` `surface.legendgrouptitle.font`: Sets this legend group's title font.

`go_surface_lighting()` `surface.lighting`

`go_surface_lightposition()` `surface.lightposition`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_surface(x = 1:3, y = c(2, 1, 3))
```

`add_table` *Table traces*

Description

Table view for detailed data viewing. The data are arranged in a grid of rows and columns. Most styling can be specified for columns, rows or individual cells. Table is using a column-major order, ie. the grid is represented as a vector of column vectors.

Usage

```
add_table(fig, ...)

## Default S3 method:
add_table(
  fig,
  ...,
  data = NULL,
  cells = NULL,
  columnorder = NULL,
  columnwidth = NULL,
  customdata = NULL,
  domain = NULL,
  header = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  ids = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  meta = NULL,
  name = NULL,
  uid = NULL,
  uirevision = NULL,
```

```
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_table(
  arg = NULL,
  ...,
  cells = NULL,
  columnorder = NULL,
  columnwidth = NULL,
  customdata = NULL,
  domain = NULL,
  header = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  ids = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  meta = NULL,
  name = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_table_cells(
  arg = NULL,
  ...,
  align = NULL,
  fill = NULL,
  font = NULL,
  format = NULL,
  height = NULL,
  line = NULL,
  prefix = NULL,
  suffix = NULL,
  values = NULL,
  skip_invalid = FALSE
)

go_table_cells_fill(arg = NULL, ..., color = NULL, skip_invalid = FALSE)

go_table_cells_font(
```

```
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_table_cells_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_table_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_table_header(
  arg = NULL,
  ...,
  align = NULL,
  fill = NULL,
  font = NULL,
  format = NULL,
  height = NULL,
  line = NULL,
  prefix = NULL,
  suffix = NULL,
  values = NULL,
  skip_invalid = FALSE
)

go_table_header_fill(arg = NULL, ..., color = NULL, skip_invalid = FALSE)
```

```
  go_table_header_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_table_header_line(
    arg = NULL,
    ...,
    color = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_table_hoverlabel(
    arg = NULL,
    ...,
    align = NULL,
    bgcolor = NULL,
    bordercolor = NULL,
    font = NULL,
    namelength = NULL,
    showarrow = NULL,
    skip_invalid = FALSE
  )

  go_table_hoverlabel_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
```

```
  skip_invalid = FALSE
)

go_table_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_table_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
cells	Named list or go_table_cells() object.
columnorder	Specifies the rendered order of the data columns; for example, a value ‘2‘ at position ‘0‘ means that column index ‘0‘ in the data will be rendered as the third column, as columns have an index base of zero. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
columnwidth	The width of columns expressed as a ratio. Columns fill the available width in proportion of their specified column widths. Number, or a vector of these.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

domain — Named list or go_table_domain() object.

header — Named list or go_table_header() object.

hoverinfo — Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of "x", "y", "z", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.

hoverlabel — Named list or go_table_hoverlabel() object.

ids — Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

legend — Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: "legend", "legend2", "legend3", ...

legendgrouptitle — Named list or go_table_legendgrouptitle() object.

legendrank — Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

legendwidth — Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

meta — Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

name — Sets the trace name. The trace name appears as the legend item and on hover. String.

uid — Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision — Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can

	add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.
visible	Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.
row	add_table(): Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells. go_table_domain(): If there is a layout grid, use the domain for this row in the grid for this table trace. Integer in the interval [0, Inf].
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
align	go_table_cells(), go_table_header(): Sets the horizontal alignment of the ‘text‘ within the box. Has an effect only if ‘text‘ spans two or more lines (i.e. ‘text‘ contains one or more HTML tags) or if an explicit width is set to override the text width. One of "left", "center", "right", or a vector of these. go_table_hoverlabel(): Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.
fill	go_table_cells(): Named list or go_table_cells_fill() object. go_table_header(): Named list or go_table_header_fill() object.
font	go_table_cells(): Named list or go_table_cells_font() object. go_table_header(): Named list or go_table_header_font() object. go_table_hoverlabel(): Sets the font used in hover labels. Named list or go_table_hoverlabel_font() object. go_table_legendgrouptitle(): Sets this legend group’s title font. Named list or go_table_legendgrouptitle_font() object.
format	Sets the cell value formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
height	The height of cells. Number.
line	go_table_cells(): Named list or go_table_cells_line() object. go_table_header(): Named list or go_table_header_line() object.
prefix	Prefix for cell values. String, or a vector of these.
suffix	Suffix for cell values. String, or a vector of these.
values	go_table_cells(): Cell values. ‘values[m][n]‘ represents the value of the ‘n‘th point in column ‘m‘, therefore the ‘values[m]‘ vector length for all columns must be the same (longer vectors will be truncated). Each value must be a finite

number or a string. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_table_header()`: Header cell values. 'values[m][n]' represents the value of the 'n'th point in column 'm', therefore the 'values[m]' vector length for all columns must be the same (longer vectors will be truncated). Each value must be a finite number or a string. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`color` `go_table_cells_fill()`, `go_table_header_fill()`: Sets the cell fill color. It accepts either a specific color or an array of colors or a 2D array of colors. Color: a CSS color string or R color name, or a vector of these.

`go_table_cells_font()`, `go_table_cells_line()`, `go_table_header_font()`, `go_table_header_line()`, `go_table_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

`go_table_legendgrouptitle_font()`: Color: a CSS color string or R color name.

`family` `go_table_cells_font()`, `go_table_header_font()`, `go_table_hoverlabel_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

`go_table_legendgrouptitle_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

`lineposition` `go_table_cells_font()`, `go_table_header_font()`, `go_table_hoverlabel_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

`go_table_legendgrouptitle_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

`shadow` `go_table_cells_font()`, `go_table_header_font()`, `go_table_hoverlabel_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

`go_table_legendgrouptitle_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

`size` `go_table_cells_font()`, `go_table_header_font()`, `go_table_hoverlabel_font()`: Number in the interval [1, Inf], or a vector of these.

`go_table_legendgrouptitle_font()`: Number in the interval [1, Inf].

style — go_table_cells_font(), go_table_header_font(), go_table_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_table_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase — go_table_cells_font(), go_table_header_font(), go_table_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_table_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant — go_table_cells_font(), go_table_header_font(), go_table_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_table_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight — go_table_cells_font(), go_table_header_font(), go_table_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_table_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

width — Number, or a vector of these.

column — If there is a layout grid, use the domain for this column in the grid for this table trace. Integer in the interval [0, Inf].

x — Sets the horizontal domain of this table trace (in plot fraction). Vector or list of 2 element(s).

y — Sets the vertical domain of this table trace (in plot fraction). Vector or list of 2 element(s).

bgcolor — Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

bordercolor — Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

namelength — Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to ‘namelength - 3‘ characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow — Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

text — Sets the title of the legend group. String.

Details

`go_table()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_table()` creates an object and adds it to a figure.

Nested objects documented here:

`go_table_cells()` `table.cells`

`go_table_cells_fill()` `table.cells.fill`

`go_table_cells_font()` `table.cells.font`

`go_table_cells_line()` `table.cells.line`

`go_table_domain()` `table.domain`

`go_table_header()` `table.header`

`go_table_header_fill()` `table.header.fill`

`go_table_header_font()` `table.header.font`

`go_table_header_line()` `table.header.line`

`go_table_hoverlabel()` `table.hoverlabel`

`go_table_hoverlabel_font()` `table.hoverlabel.font`: Sets the font used in hover labels.

`go_table_legendgrouptitle()` `table.legendgrouptitle`

`go_table_legendgrouptitle_font()` `table.legendgrouptitle.font`: Sets this legend group's title font.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_table()
```

add_traces *Add traces to a figure*

Description

add_trace() adds one trace and add_traces() several. add_trace() is an S3 generic, so packages can add methods for their own figure classes (the plotly package does for plot_ly() objects).

Usage

```
add_trace(fig, ...)

## S3 method for class 'go_figure'
add_trace(
  fig,
  trace,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  exclude_empty_subplots = FALSE,
  ...
)

add_traces(
  fig,
  data,
  rows = NULL,
  cols = NULL,
  secondary_ys = NULL,
  exclude_empty_subplots = FALSE
)
```

Arguments

fig	A go_figure object.
...	Unused.
trace	A trace: a graph object such as go_scatter(), or a named list with a type element (default "scatter").
row, col	Subplot grid cell for the trace (figures created with make_subplots() only). Either may be "all" or a vector, to add the trace to several cells.
secondary_y	If TRUE, use the secondary y axis of the subplot (see the specs argument of make_subplots()).
exclude_empty_subplots	If TRUE, don't add the trace to subplots that have no traces or layout objects yet.
data	A list of traces.

rows, cols, secondary_ys
: Per-trace grid cells and secondary y flags for add_traces(); a single value applies to every trace.

Value

The modified figure.

See Also

The generated add_<trace type>() functions, e.g. add_scatter().

Examples

```
go_figure() |>
  add_trace(go_scatter(x = 1:3, y = 1:3)) |>
  add_traces(list(list(type = "scatter", y = 3:1), go_scatter(y = c(2, 2, 2))))

make_subplots(rows = 1, cols = 2) |>
  add_trace(go_scatter(y = 1:3), row = 1, col = 2)
```

---

add_treemap *Treemap traces*

Description

Visualize hierarchal data from leaves (and/or outer branches) towards root with rectangles. The treemap sectors are determined by the entries in "labels" or "ids" and in "parents".

Usage

```
add_treemap(fig, ...)

## Default S3 method:
add_treemap(
  fig,
  ...,
  data = NULL,
  branchvalues = NULL,
  count = NULL,
  customdata = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
```

```
  insidetextfont = NULL,
  labels = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  level = NULL,
  marker = NULL,
  maxdepth = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  outsidetextfont = NULL,
  parents = NULL,
  pathbar = NULL,
  root = NULL,
  sort = NULL,
  text = NULL,
  textfont = NULL,
  textinfo = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  tiling = NULL,
  uid = NULL,
  uirevision = NULL,
  values = NULL,
  visible = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_treemap(
  arg = NULL,
  ...,
  branchvalues = NULL,
  count = NULL,
  customdata = NULL,
  domain = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextfont = NULL,
  labels = NULL,
```

```
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  level = NULL,
  marker = NULL,
  maxdepth = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  outsidetextfont = NULL,
  parents = NULL,
  pathbar = NULL,
  root = NULL,
  sort = NULL,
  text = NULL,
  textfont = NULL,
  textinfo = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  tiling = NULL,
  uid = NULL,
  uirevision = NULL,
  values = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_treemap_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_treemap_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
```

```
  skip_invalid = FALSE
)

go_treemap_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_treemap_insidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_treemap_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_treemap_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
```

```
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_treemap_marker(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colors = NULL,
  colorscale = NULL,
  cornerradius = NULL,
  depthfade = NULL,
  line = NULL,
  pad = NULL,
  pattern = NULL,
  reversescale = NULL,
  showscale = NULL,
  skip_invalid = FALSE
)

go_treemap_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
```

```
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_treemap_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

```
  go_treemap_marker_colorbar_tickformatstop(
    arg = NULL,
    ...,
    dtickrange = NULL,
    enabled = NULL,
    name = NULL,
    templateitemname = NULL,
    value = NULL,
    skip_invalid = FALSE
  )

  go_treemap_marker_colorbar_title(
    arg = NULL,
    ...,
    font = NULL,
    side = NULL,
    text = NULL,
    skip_invalid = FALSE
  )

  go_treemap_marker_colorbar_title_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_treemap_marker_line(
    arg = NULL,
    ...,
    color = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_treemap_marker_pad(
    arg = NULL,
    ...,
    b = NULL,
```

```
  l = NULL,
  r = NULL,
  t = NULL,
  skip_invalid = FALSE
)

go_treemap_marker_pattern(
  arg = NULL,
  ...,
  bgcolor = NULL,
  fgcolor = NULL,
  fgopacity = NULL,
  fillmode = NULL,
  path = NULL,
  shape = NULL,
  size = NULL,
  solidity = NULL,
  skip_invalid = FALSE
)

go_treemap_outsidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_treemap_pathbar(
  arg = NULL,
  ...,
  edgeshape = NULL,
  side = NULL,
  textfont = NULL,
  thickness = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_treemap_pathbar_textfont(
  arg = NULL,
```

```
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_treemap_root(arg = NULL, ..., color = NULL, skip_invalid = FALSE)

go_treemap_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_treemap_tiling(
  arg = NULL,
  ...,
  flip = NULL,
  packing = NULL,
  pad = NULL,
  squarifyratio = NULL,
  skip_invalid = FALSE
)
```

Arguments

`fig`	A `go_figure` object.
`...`	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
`data`	A data frame in which formula properties (e.g. `x = ~col`, `marker_size = ~col`) are evaluated. Default: the data given to `go_figure()`. Only the evaluated

values are added to the trace. A `crosstalk::SharedData` works the same way, and links the trace to its crosstalk group (see `highlight()`).

`branchvalues` Determines how the items in ‘values‘ are summed. When set to "total", items in ‘values‘ are taken to be value of all its descendants. When set to "remainder", items in ‘values‘ corresponding to the root and the branches sectors are taken to be the extra part not part of the sum of the values at their leaves. One of `"remainder"`, `"total"`.

`count` Determines default for ‘values‘ when it is not provided, by inferring a 1 for each of the "leaves" and/or "branches", otherwise 0. Any combination of `"branches"`, `"leaves"` joined with `"+"`.

`customdata` Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`domain` Named list or `go_treemap_domain()` object.

`hoverinfo` Determines what trace information appears on hover. Flags are rendered in a fixed order; use ‘hovertemplate‘ if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of `"label"`, `"text"`, `"value"`, `"name"`, `"current path"`, `"percent root"`, `"percent entry"`, `"percent parent"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_treemap_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variables ‘currentPath‘, ‘root‘, ‘entry‘, ‘percentRoot‘, ‘percentEntry‘ and ‘percentParent‘. Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example ‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each sector. If a single string, the same string appears for all data points. If an array of string, the items are mapped in order of this trace's sectors. To be seen, trace ‘hoverinfo‘ must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`insidetextfont`
: Sets the font used for ‘textinfo‘ lying inside the sector. Named list or `go_treemap_insidetextfont()` object.

`labels`
: Sets the labels of each of the sectors. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgrouptitle`
: Named list or `go_treemap_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`level`
: Sets the level from which this trace hierarchy is rendered. Set ‘level‘ to ‘”‘ to start from the root node in the hierarchy. Must be an "id" if ‘ids‘ is filled in, otherwise plotly attempts to find a matching item in ‘labels‘. Any value.

`marker`
: Named list or `go_treemap_marker()` object.

`maxdepth`
: Sets the number of rendered sectors from any given ‘level‘. Set ‘maxdepth‘ to "-1" to render all the levels in the hierarchy. Integer.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

name
: go_treemap(): Sets the trace name. The trace name appears as the legend item and on hover. String.

  go_treemap_marker_colorbar_tickformatstop(): When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

opacity
: Sets the opacity of the trace. Number in the interval [0, 1].

outsidetextfont
: Sets the font used for ‘textinfo‘ lying outside the sector. This option refers to the root of the hierarchy presented on top left corner of a treemap graph. Please note that if a hierarchy has multiple root nodes, this option won't have any effect and ‘insidetextfont‘ would be used. Named list or go_treemap_outsidetextfont() object.

parents
: Sets the parent sectors for each of the sectors. Empty string items '' are understood to reference the root node in the hierarchy. If ‘ids‘ is filled, ‘parents‘ items are understood to be "ids" themselves. When ‘ids‘ is not set, plotly attempts to find matching items in ‘labels‘, but beware they must be unique. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

pathbar
: Named list or go_treemap_pathbar() object.

root
: Named list or go_treemap_root() object.

sort
: Determines whether or not the sectors are reordered from largest to smallest. Logical.

text
: go_treemap(): Sets text elements associated with each sector. If trace ‘textinfo‘ contains a "text" flag, these elements will be seen on the chart. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

  go_treemap_legendgrouptitle(): Sets the title of the legend group. String.

  go_treemap_marker_colorbar_title(): Sets the title of the color bar. String.

textfont
: go_treemap(): Sets the font used for ‘textinfo‘. Named list or go_treemap_textfont() object.

  go_treemap_pathbar(): Sets the font used inside ‘pathbar‘. Named list or go_treemap_pathbar_textfont() object.

textinfo
: Determines what trace information appears on the graph. Flags are rendered in a fixed order; use ‘texttemplate‘ if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of "label", "text", "value", "current path", "percent root", "percent entry", "percent parent" joined with "+", or one of "none".

textposition
: Sets the positions of the ‘text‘ elements. One of "top left", "top center", "top right", "middle left", "middle center", "middle right", "bottom left", "bottom center", "bottom right".

texttemplate
: Template string used for rendering the information text that appears on points. Note that this will override ‘textinfo‘. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax

%{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'currentPath', 'root', 'entry', 'percentRoot', 'percentEntry', 'percentParent', 'label' and 'value'. String, or a vector of these.

`texttemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`tiling`
: Named list or `go_treemap_tiling()` object.

`uid`
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision`
: Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

`values`
: Sets the values associated with each of the sectors. Use with 'branchvalues' to determine how the values are summed. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`visible`
: `go_treemap()`: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

  `go_treemap_pathbar()`: Determines if the path bar is drawn i.e. outside the trace 'domain' and with one pixel gap. Logical.

`row`
: `add_treemap()`: Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

  `go_treemap_domain()`: If there is a layout grid, use the domain for this row in the grid for this treemap trace. Integer in the interval [0, Inf].

`col`
: Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`skip_invalid`
: If `TRUE`, silently drop invalid properties instead of raising an error.

`arg` Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

`column` If there is a layout grid, use the domain for this column in the grid for this treemap trace. Integer in the interval [0, Inf].

`x` `go_treemap_domain()`: Sets the horizontal domain of this treemap trace (in plot fraction). Vector or list of 2 element(s).

`go_treemap_marker_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`y` `go_treemap_domain()`: Sets the vertical domain of this treemap trace (in plot fraction). Vector or list of 2 element(s).

`go_treemap_marker_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

`align` Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`bgcolor` `go_treemap_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`go_treemap_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

`go_treemap_marker_pattern()`: When there is no colorscale sets the color of background pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "overlay". Otherwise, defaults to a transparent background. Color: a CSS color string or R color name, or a vector of these.

`bordercolor` `go_treemap_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`go_treemap_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

`font` `go_treemap_hoverlabel()`: Sets the font used in hover labels. Named list or `go_treemap_hoverlabel_font()` object.

`go_treemap_legendgrouptitle()`: Sets this legend group’s title font. Named list or `go_treemap_legendgrouptitle_font()` object.

`go_treemap_marker_colorbar_title()`: Sets this color bar’s title font. Named list or `go_treemap_marker_colorbar_title_font()` object.

`namelength` Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to ‘namelength - 3‘

characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow — Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color — go_treemap_hoverlabel_font(), go_treemap_insidetextfont(), go_treemap_outsidetextfont(), go_treemap_pathbar_textfont(), go_treemap_textfont(): Color: a CSS color string or R color name, or a vector of these.

go_treemap_legendgrouptitle_font(), go_treemap_marker_colorbar_tickfont(), go_treemap_marker_colorbar_title_font(): Color: a CSS color string or R color name.

go_treemap_marker_line(): Sets the color of the line enclosing each sector. Defaults to the 'paper_bgcolor' value. Color: a CSS color string or R color name, or a vector of these.

go_treemap_root(): sets the color of the root node for a sunburst/treemap/icicle trace. this has no effect when a colorscale is used to set the markers. Color: a CSS color string or R color name.

family — go_treemap_hoverlabel_font(), go_treemap_insidetextfont(), go_treemap_outsidetextfont(), go_treemap_pathbar_textfont(), go_treemap_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

go_treemap_legendgrouptitle_font(), go_treemap_marker_colorbar_tickfont(), go_treemap_marker_colorbar_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition — go_treemap_hoverlabel_font(), go_treemap_insidetextfont(), go_treemap_outsidetextfont(), go_treemap_pathbar_textfont(), go_treemap_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_treemap_legendgrouptitle_font(), go_treemap_marker_colorbar_tickfont(), go_treemap_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow — go_treemap_hoverlabel_font(), go_treemap_insidetextfont(), go_treemap_outsidetextfont(), go_treemap_pathbar_textfont(), go_treemap_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_treemap_legendgrouptitle_font(), go_treemap_marker_colorbar_tickfont(), go_treemap_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text

font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: go_treemap_hoverlabel_font(), go_treemap_insidetextfont(), go_treemap_outsidetextfont(), go_treemap_pathbar_textfont(), go_treemap_textfont(): Number in the interval [1, Inf], or a vector of these.

go_treemap_legendgrouptitle_font(), go_treemap_marker_colorbar_tickfont(), go_treemap_marker_colorbar_title_font(): Number in the interval [1, Inf].

go_treemap_marker_pattern(): Sets the size of unit squares of the pattern fill in pixels, which corresponds to the interval of repetition of the pattern. Number in the interval [0, Inf], or a vector of these.

style
: go_treemap_hoverlabel_font(), go_treemap_insidetextfont(), go_treemap_outsidetextfont(), go_treemap_pathbar_textfont(), go_treemap_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_treemap_legendgrouptitle_font(), go_treemap_marker_colorbar_tickfont(), go_treemap_marker_colorbar_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_treemap_hoverlabel_font(), go_treemap_insidetextfont(), go_treemap_outsidetextfont(), go_treemap_pathbar_textfont(), go_treemap_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_treemap_legendgrouptitle_font(), go_treemap_marker_colorbar_tickfont(), go_treemap_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_treemap_hoverlabel_font(), go_treemap_insidetextfont(), go_treemap_outsidetextfont(), go_treemap_pathbar_textfont(), go_treemap_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_treemap_legendgrouptitle_font(), go_treemap_marker_colorbar_tickfont(), go_treemap_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_treemap_hoverlabel_font(), go_treemap_insidetextfont(), go_treemap_outsidetextfont(), go_treemap_pathbar_textfont(), go_treemap_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_treemap_legendgrouptitle_font(), go_treemap_marker_colorbar_tickfont(), go_treemap_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

autocolorscale
: Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if colors is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

cauto — Determines whether or not the color domain is computed with respect to the input data (here colors) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if colors is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

cmax — Sets the upper bound of the color domain. Has an effect only if colors is set to a numerical array. Value should have the same units as colors and if set, ‘marker.cmin‘ must be set as well. Number.

cmid — Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if colors is set to a numerical array. Value should have the same units as colors. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

cmin — Sets the lower bound of the color domain. Has an effect only if colors is set to a numerical array. Value should have the same units as colors and if set, ‘marker.cmax‘ must be set as well. Number.

coloraxis — Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

colorbar — Named list or `go_treemap_marker_colorbar()` object.

colors — Sets the color of each sector of this trace. If not specified, the default trace color set is used to pick the sector colors. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

colorscale — Sets the colorscale. Has an effect only if colors is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

cornerradius — Sets the maximum rounding of corners (in px). Number in the interval [0, Inf].

depthfade — Determines if the sector colors are faded towards the background from the leaves up to the headers. This option is unavailable when a ‘colorscale‘ is present, defaults to false when ‘marker.colors‘ is set, but otherwise defaults to true. When set to "reversed", the fading direction is inverted, that is the top elements within hierarchy are drawn with fully saturated colors while the leaves are faded towards the background color. One of `"reversed"`, `TRUE`, `FALSE`.

line — Named list or `go_treemap_marker_line()` object.

pad — `go_treemap_marker()`: Named list or `go_treemap_marker_pad()` object.

`go_treemap_tiling()`: Sets the inner padding (in px). Number in the interval [0, Inf].

`pattern` Sets the pattern within the marker. Named list or `go_treemap_marker_pattern()` object.

`reversescale` Reverses the color mapping if true. Has an effect only if colors is set to a numerical array. If true, ‘marker.cmin‘ will correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if colors is set to a numerical array. Logical.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticksuffix Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

thickness `go_treemap_marker_colorbar()`: Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`go_treemap_pathbar()`: Sets the thickness of 'pathbar' (in px). If not specified the 'pathbar.textfont.size' is used with 3 pixles extra padding on each side. Number in the interval [12, Inf].

thicknessmode Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

tick0 Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar's tick label font Named list or `go_treemap_marker_colorbar_tickfont()` object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or `go_treemap_marker_colorbar_tickformatstop()` objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition`
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep`
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen`
: Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode`
: Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix`
: Sets a tick label prefix. String.

`ticks`
: Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix`
: Sets a tick label suffix. String.

`ticktext`
: Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals`
: Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth`
: Sets the tick width (in px). Number in the interval [0, Inf].

`title`
: String (the title text), named list or `go_treemap_marker_colorbar_title()` object.

`xanchor`
: Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad`
: Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref`
: Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`yanchor`
: Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

side — `go_treemap_marker_colorbar_title()`: Determines the location of color bar’s title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.

`go_treemap_pathbar()`: Determines on which side of the treemap the ‘pathbar‘ should be presented. One of `"top"`, `"bottom"`.

width — Sets the width (in px) of the line enclosing each sector. Number in the interval [0, Inf], or a vector of these.

b — Sets the padding form the bottom (in px). Number in the interval [0, Inf].

l — Sets the padding form the left (in px). Number in the interval [0, Inf].

r — Sets the padding form the right (in px). Number in the interval [0, Inf].

t — Sets the padding form the top (in px). Number in the interval [0, Inf].

fgcolor — When there is no colorscale sets the color of foreground pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "replace". Otherwise, defaults to dark grey or white to increase contrast with the ‘bgcolor‘. Color: a CSS color string or R color name, or a vector of these.

fgopacity — Sets the opacity of the foreground pattern fill. Defaults to a 0.5 when ‘fillmode‘ is "overlay". Otherwise, defaults to 1. Number in the interval [0, 1].

fillmode — Determines whether ‘marker.color‘ should be used as a default to ‘bgcolor‘ or a ‘fgcolor‘. One of `"replace"`, `"overlay"`.

path — Sets a custom path for pattern fill. Use with no ‘shape‘ or ‘solidity‘, provide an SVG path string for the regions of the square from (0,0) to (‘size‘,‘size‘) to color. String, or a vector of these.

shape — Sets the shape of the pattern fill. By default, no pattern is used for filling the area. One of `""`, `"/"`, `"\\"`, `"x"`, `"-"`, `"|"`, `"+"`, `"."`, or a vector of these.

`solidity` Sets the solidity of the pattern fill. Solidity is roughly the fraction of the area filled by the pattern. Solidity of 0 shows only the background color without pattern and solidty of 1 shows only the foreground color without pattern. Number in the interval [0, 1], or a vector of these.

`edgeshape` Determines which shape is used for edges between 'barpath' labels. One of `">"`, `"<"`, `"|"`, `"/"`, `"\\"`.

`flip` Determines if the positions obtained from solver are flipped on each axis. Any combination of `"x"`, `"y"` joined with `"+"`.

`packing` Determines d3 treemap solver. For more info please refer to https://github.com/d3/d3-hierarchy#treemap-tiling One of `"squarify"`, `"binary"`, `"dice"`, `"slice"`, `"slice-dice"`, `"dice-slice"`.

`squarifyratio` When using "squarify" 'packing' algorithm, according to https://github.com/d3/d3-hierarchy/blob/v3.1.1/README.md#squarify_ratio this option specifies the desired aspect ratio of the generated rectangles. The ratio must be specified as a number greater than or equal to one. Note that the orientation of the generated rectangles (tall or wide) is not implied by the ratio; for example, a ratio of two will attempt to produce a mixture of rectangles whose width:height ratio is either 2:1 or 1:2. When using "squarify", unlike d3 which uses the Golden Ratio i.e. 1.618034, Plotly applies 1 to increase squares in treemap layouts. Number in the interval [1, Inf].

Details

`go_treemap()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_treemap()` creates an object and adds it to a figure.

Nested objects documented here:

`go_treemap_domain()` `treemap.domain`

`go_treemap_hoverlabel()` `treemap.hoverlabel`

`go_treemap_hoverlabel_font()` `treemap.hoverlabel.font`: Sets the font used in hover labels.

`go_treemap_insidetextfont()` `treemap.insidetextfont`: Sets the font used for 'textinfo' lying inside the sector.

`go_treemap_legendgrouptitle()` `treemap.legendgrouptitle`

`go_treemap_legendgrouptitle_font()` `treemap.legendgrouptitle.font`: Sets this legend group's title font.

`go_treemap_marker()` `treemap.marker`

`go_treemap_marker_colorbar()` `treemap.marker.colorbar`

`go_treemap_marker_colorbar_tickfont()` `treemap.marker.colorbar.tickfont`: Sets the color bar's tick label font

`go_treemap_marker_colorbar_tickformatstop()` `treemap.marker.colorbar.tickformatstop`

`go_treemap_marker_colorbar_title()` `treemap.marker.colorbar.title`

`go_treemap_marker_colorbar_title_font()` `treemap.marker.colorbar.title.font`: Sets this color bar's title font.

go_treemap_marker_line() treemap.marker.line

go_treemap_marker_pad() treemap.marker.pad

go_treemap_marker_pattern() treemap.marker.pattern: Sets the pattern within the marker.

go_treemap_outsidetextfont() treemap.outsidetextfont: Sets the font used for 'textinfo' lying outside the sector. This option refers to the root of the hierarchy presented on top left corner of a treemap graph. Please note that if a hierarchy has multiple root nodes, this option won't have any effect and 'insidetextfont' would be used.

go_treemap_pathbar() treemap.pathbar

go_treemap_pathbar_textfont() treemap.pathbar.textfont: Sets the font used inside 'pathbar'.

go_treemap_root() treemap.root

go_treemap_textfont() treemap.textfont: Sets the font used for 'textinfo'.

go_treemap_tiling() treemap.tiling

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_treemap()
```

---

add_violin *Violin traces*

Description

In vertical (horizontal) violin plots, statistics are computed using 'y' ('x') values. By supplying an 'x' ('y') array, one violin per distinct x (y) value is drawn If no 'x' ('y') vector is provided, a single violin is drawn. That violin position is then positioned with 'name' or with 'x0' ('y0') if provided.

Usage

```
add_violin(fig, ...)

## Default S3 method:
add_violin(
  fig,
  ...,
  data = NULL,
```

```
alignmentgroup = NULL,
bandwidth = NULL,
box = NULL,
customdata = NULL,
fillcolor = NULL,
hoverinfo = NULL,
hoverlabel = NULL,
hoveron = NULL,
hovertemplate = NULL,
hovertemplatefallback = NULL,
hovertext = NULL,
ids = NULL,
jitter = NULL,
legend = NULL,
legendgroup = NULL,
legendgrouptitle = NULL,
legendrank = NULL,
legendwidth = NULL,
line = NULL,
marker = NULL,
meanline = NULL,
meta = NULL,
name = NULL,
offsetgroup = NULL,
opacity = NULL,
orientation = NULL,
pointpos = NULL,
points = NULL,
quartilemethod = NULL,
scalegroup = NULL,
scalemode = NULL,
selected = NULL,
selectedpoints = NULL,
showlegend = NULL,
side = NULL,
span = NULL,
spanmode = NULL,
text = NULL,
uid = NULL,
uirevision = NULL,
unselected = NULL,
visible = NULL,
width = NULL,
x = NULL,
x0 = NULL,
xaxis = NULL,
xhoverformat = NULL,
y = NULL,
```

```
  y0 = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  zorder = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_violin(
  arg = NULL,
  ...,
  alignmentgroup = NULL,
  bandwidth = NULL,
  box = NULL,
  customdata = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  jitter = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  marker = NULL,
  meanline = NULL,
  meta = NULL,
  name = NULL,
  offsetgroup = NULL,
  opacity = NULL,
  orientation = NULL,
  pointpos = NULL,
  points = NULL,
  quartilemethod = NULL,
  scalegroup = NULL,
  scalemode = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  side = NULL,
```

```
  span = NULL,
  spanmode = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  width = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xhoverformat = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  zorder = NULL,
  skip_invalid = FALSE
)

go_violin_box(
  arg = NULL,
  ...,
  fillcolor = NULL,
  line = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_violin_box_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_violin_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
```

```
)

go_violin_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_violin_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_violin_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_violin_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)
```

```
  go_violin_marker(
    arg = NULL,
    ...,
    angle = NULL,
    color = NULL,
    line = NULL,
    opacity = NULL,
    outliercolor = NULL,
    size = NULL,
    symbol = NULL,
    skip_invalid = FALSE
  )

  go_violin_marker_line(
    arg = NULL,
    ...,
    color = NULL,
    outliercolor = NULL,
    outlierwidth = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_violin_meanline(
    arg = NULL,
    ...,
    color = NULL,
    visible = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_violin_selected(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

  go_violin_selected_marker(
    arg = NULL,
    ...,
    color = NULL,
    opacity = NULL,
    size = NULL,
    skip_invalid = FALSE
  )

  go_violin_unselected(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

  go_violin_unselected_marker(
    arg = NULL,
    ...,
```

```
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
alignmentgroup	Set several traces linked to the same position axis or matching axes to the same alignmentgroup. This controls whether bars compute their positional range dependently or independently. String.
bandwidth	Sets the bandwidth used to compute the kernel density estimate. By default, the bandwidth is determined by Silverman's rule of thumb. Number in the interval [0, Inf].
box	Named list or go_violin_box() object.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
fillcolor	go_violin(): Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name. go_violin_box(): Sets the inner box plot fill color. Color: a CSS color string or R color name.
hoverinfo	Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of "x", "y", "z", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_violin_hoverlabel() object.
hoveron	Do the hover effects highlight individual violins or sample points or the kernel density estimate or any combination of them? Any combination of "violins", "points", "kde" joined with "+", or one of "all".
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown.

Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Same as 'text'. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`jitter`
: Sets the amount of jitter in the sample points drawn. If 0, the sample points align along the distribution axis. If 1, the sample points are drawn in a random jitter of width equal to the width of the violins. Number in the interval [0, 1].

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_violin_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line`
: `go_violin()`: Named list or `go_violin_line()` object.
`go_violin_box()`: Named list or `go_violin_box_line()` object.
`go_violin_marker()`: Named list or `go_violin_marker_line()` object.

marker
: go_violin(): Named list or go_violin_marker() object.
go_violin_selected(): Named list or go_violin_selected_marker() object.
go_violin_unselected(): Named list or go_violin_unselected_marker() object.

meanline
: Named list or go_violin_meanline() object.

meta
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

name
: Sets the trace name. The trace name appears as the legend item and on hover. For violin traces, the name will also be used for the position coordinate, if ‘x‘ and ‘x0‘ (‘y‘ and ‘y0‘ if horizontal) are missing and the position axis is categorical. Note that the trace name is also used as a default value for attribute ‘scalegroup‘ (please see its description for details). String.

offsetgroup
: Set several traces linked to the same position axis or matching axes to the same offsetgroup where bars of the same position coordinate will line up. String.

opacity
: go_violin(): Sets the opacity of the trace. Number in the interval [0, 1].
go_violin_marker(): Sets the marker opacity. Number in the interval [0, 1].
go_violin_selected_marker(): Sets the marker opacity of selected points. Number in the interval [0, 1].
go_violin_unselected_marker(): Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

orientation
: Sets the orientation of the violin(s). If "v" ("h"), the distribution is visualized along the vertical (horizontal). One of "v", "h".

pointpos
: Sets the position of the sample points in relation to the violins. If 0, the sample points are places over the center of the violins. Positive (negative) values correspond to positions to the right (left) for vertical violins and above (below) for horizontal violins. Number in the interval [-2, 2].

points
: If "outliers", only the sample points lying outside the whiskers are shown If "suspectedoutliers", the outlier points are shown and points either less than 4"Q1-3"Q3 or greater than 4"Q3-3"Q1 are highlighted (see ‘outliercolor‘) If "all", all sample points are shown If FALSE, only the violins are shown with no sample points. Defaults to "suspectedoutliers" when ‘marker.outliercolor‘ or ‘marker.line.outliercolor‘ is set, otherwise defaults to "outliers". One of "all", "outliers", "suspectedoutliers", FALSE.

quartilemethod
: Sets the method used to compute the sample’s Q1 and Q3 quartiles. The "linear" method uses the 25th percentile for Q1 and 75th percentile for Q3 as computed using method #10 (listed on http://jse.amstat.org/v14n3/langford.html). The "exclusive" method uses the median to divide the ordered dataset into two halves if the sample is odd, it does not include the median in either half - Q1

is then the median of the lower half and Q3 the median of the upper half. The "inclusive" method also uses the median to divide the ordered dataset into two halves but if the sample is odd, it includes the median in both halves - Q1 is then the median of the lower half and Q3 the median of the upper half. One of `"linear"`, `"exclusive"`, `"inclusive"`.

`scalegroup` If there are multiple violins that should be sized according to some metric (see 'scalemode'), link them by providing a non-empty group id here shared by every trace in the same group. If a violin's 'width' is undefined, 'scalegroup' will default to the trace's name. In this case, violins with the same names will be linked together String.

`scalemode` Sets the metric by which the width of each violin is determined. "width" means each violin has the same (max) width "count" means the violins are scaled by the number of sample points making up each violin. One of `"width"`, `"count"`.

`selected` Named list or `go_violin_selected()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the 'unselected' are turned on for all points, whereas, any other non-array values means no selection all where the 'selected' and 'unselected' styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`side` Determines on which side of the position value the density function making up one half of a violin is plotted. Useful when comparing two violin traces under "overlay" mode, where one trace has 'side' set to "positive" and the other to "negative". One of `"both"`, `"positive"`, `"negative"`.

`span` Sets the span in data space for which the density function will be computed. Has an effect only when 'spanmode' is set to "manual". Vector or list of 2 element(s).

`spanmode` Sets the method by which the span in data space where the density function will be computed. "soft" means the span goes from the sample's minimum value minus two bandwidths to the sample's maximum value plus two bandwidths. "hard" means the span goes from the sample's minimum to its maximum value. For custom span settings, use mode "manual" and fill in the 'span' attribute. One of `"soft"`, `"hard"`, `"manual"`.

`text` `go_violin()`: Sets the text elements associated with each sample value. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`go_violin_legendgrouptitle()`: Sets the title of the legend group. String.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable:

	true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.
`unselected`	Named list or `go_violin_unselected()` object.
`visible`	`go_violin()`: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`. `go_violin_box()`: Determines if an miniature box plot is drawn inside the violins. Logical. `go_violin_meanline()`: Determines if a line corresponding to the sample's mean is shown inside the violins. If ‘box.visible‘ is turned on, the mean line is drawn inside the inner box. Otherwise, the mean line is drawn from one side of the violin to other. Logical.
`width`	`go_violin()`: Sets the width of the violin in data coordinates. If 0 (default value) the width is automatically selected based on the positions of other violin traces in the same subplot. Number in the interval [0, Inf]. `go_violin_box()`: Sets the width of the inner box plots relative to the violins' width. For example, with 1, the inner box plots are as wide as the violins. Number in the interval [0, 1]. `go_violin_box_line()`: Sets the inner box plot bounding line width. Number in the interval [0, Inf]. `go_violin_line()`: Sets the width (in px) of line bounding the violin(s). Number in the interval [0, Inf]. `go_violin_marker_line()`: Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf]. `go_violin_meanline()`: Sets the mean line width. Number in the interval [0, Inf].
`x`	Sets the x sample data or coordinates. See overview for more info. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
`x0`	Sets the x coordinate for single-box traces or the starting coordinate for multi-box traces set using q1/median/q3. See overview for more info. Any value.
`xaxis`	Sets a reference between this trace's x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...
`xhoverformat`	Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.
`y`	Sets the y sample data or coordinates. See overview for more info. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

y0	Sets the y coordinate for single-box traces or the starting coordinate for multi-box traces set using q1/median/q3. See overview for more info. Any value.
yaxis	Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to 'layout.yaxis'. If "y2", the y coordinates refer to 'layout.yaxis2', and so on. Subplot id: "y", "y2", "y3", ...
yhoverformat	Sets the hover text formatting rule for 'y' using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using 'yaxis.hoverformat'. String.
zorder	Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher 'zorder' appear in front of those with lower 'zorder'. Integer.
row	Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
secondary_y	If TRUE, use the secondary y axis of the subplot.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
color	go_violin_box_line(): Sets the inner box plot bounding line color. Color: a CSS color string or R color name. go_violin_hoverlabel_font(): Color: a CSS color string or R color name, or a vector of these. go_violin_legendgrouptitle_font(): Color: a CSS color string or R color name. go_violin_line(): Sets the color of line bounding the violin(s). Color: a CSS color string or R color name. go_violin_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.cmin' and 'marker.cmax' if set. Color: a CSS color string or R color name. go_violin_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.line.cmin' and 'marker.line.cmax' if set. Color: a CSS color string or R color name. go_violin_meanline(): Sets the mean line color. Color: a CSS color string or R color name. go_violin_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name. go_violin_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

`align` Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`bgcolor` Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`bordercolor` Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`font` `go_violin_hoverlabel()`: Sets the font used in hover labels. Named list or `go_violin_hoverlabel_font()` object.

`go_violin_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_violin_legendgrouptitle_font()` object.

`namelength` Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

`showarrow` Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

`family` `go_violin_hoverlabel_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

`go_violin_legendgrouptitle_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

`lineposition` `go_violin_hoverlabel_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

`go_violin_legendgrouptitle_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

`shadow` `go_violin_hoverlabel_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

`go_violin_legendgrouptitle_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: go_violin_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.
go_violin_legendgrouptitle_font(): Number in the interval [1, Inf].
go_violin_marker(): Sets the marker size (in px). Number in the interval [0, Inf].
go_violin_selected_marker(): Sets the marker size of selected points. Number in the interval [0, Inf].
go_violin_unselected_marker(): Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

style
: go_violin_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.
go_violin_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_violin_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.
go_violin_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_violin_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.
go_violin_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_violin_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.
go_violin_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

angle
: Sets the marker angle in respect to ‘angleref‘. Angle in degrees.

outliercolor
: go_violin_marker(): Sets the color of the outlier sample points. Color: a CSS color string or R color name.
go_violin_marker_line(): Sets the border line color of the outlier sample points. Defaults to marker.color Color: a CSS color string or R color name.

symbol
: Sets the marker symbol type. Adding 100 is equivalent to appending "-open" to a symbol name. Adding 200 is equivalent to appending "-dot" to a symbol name. Adding 300 is equivalent to appending "-open-dot" or "dot-open" to a symbol name. One of "0", "circle", "100", "circle-open", "200", "circle-dot", "300", "circle-open-dot", "1", "square", "101", "square-open", "201", "square-dot", "301", "square-open-dot", "2", "diamond", "102", "diamond-open", "202", "diamond-dot", "302", "diamond-open-dot", "3", "cross", "103", "cross-open", "203", "cross-dot", "303", "cross-open-dot", "4", "x", "104", "x-open", "204", "x-dot", "304", "x-open-dot", "5", "triangle-up", "105", "triangle-up-open", "205", "triangle-up-dot", "305", "triangle-up-open-dot",

"6", "triangle-down", "106", "triangle-down-open", "206", "triangle-down-dot",
"306", "triangle-down-open-dot", "7", "triangle-left", "107", "triangle-left-open",
"207", "triangle-left-dot", "307", "triangle-left-open-dot", "8", "triangle-right",
"108", "triangle-right-open", "208", "triangle-right-dot", "308", "triangle-right-open-do
"9", "triangle-ne", "109", "triangle-ne-open", "209", "triangle-ne-dot",
"309", "triangle-ne-open-dot", "10", "triangle-se", "110", "triangle-se-open",
"210", "triangle-se-dot", "310", "triangle-se-open-dot", "11", "triangle-sw",
"111", "triangle-sw-open", "211", "triangle-sw-dot", "311", "triangle-sw-open-dot",
"12", "triangle-nw", "112", "triangle-nw-open", "212", "triangle-nw-dot",
"312", "triangle-nw-open-dot", "13", "pentagon", "113", "pentagon-open",
"213", "pentagon-dot", "313", "pentagon-open-dot", "14", "hexagon",
"114", "hexagon-open", "214", "hexagon-dot", "314", "hexagon-open-dot",
"15", "hexagon2", "115", "hexagon2-open", "215", "hexagon2-dot", "315",
"hexagon2-open-dot", "16", "octagon", "116", "octagon-open", "216",
"octagon-dot", "316", "octagon-open-dot", "17", "star", "117", "star-open",
"217", "star-dot", "317", "star-open-dot", "18", "hexagram", "118", "hexagram-open",
"218", "hexagram-dot", "318", "hexagram-open-dot", "19", "star-triangle-up",
"119", "star-triangle-up-open", "219", "star-triangle-up-dot", "319",
"star-triangle-up-open-dot", "20", "star-triangle-down", "120", "star-triangle-down-open
"220", "star-triangle-down-dot", "320", "star-triangle-down-open-dot",
"21", "star-square", "121", "star-square-open", "221", "star-square-dot",
"321", "star-square-open-dot", "22", "star-diamond", "122", "star-diamond-open",
"222", "star-diamond-dot", "322", "star-diamond-open-dot", "23", "diamond-tall",
"123", "diamond-tall-open", "223", "diamond-tall-dot", "323", "diamond-tall-open-dot",
"24", "diamond-wide", "124", "diamond-wide-open", "224", "diamond-wide-dot",
"324", "diamond-wide-open-dot", "25", "hourglass", "125", "hourglass-open",
"26", "bowtie", "126", "bowtie-open", "27", "circle-cross", "127", "circle-cross-open",
"28", "circle-x", "128", "circle-x-open", "29", "square-cross", "129",
"square-cross-open", "30", "square-x", "130", "square-x-open", "31",
"diamond-cross", "131", "diamond-cross-open", "32", "diamond-x", "132",
"diamond-x-open", "33", "cross-thin", "133", "cross-thin-open", "34",
"x-thin", "134", "x-thin-open", "35", "asterisk", "135", "asterisk-open",
"36", "hash", "136", "hash-open", "236", "hash-dot", "336", "hash-open-dot",
"37", "y-up", "137", "y-up-open", "38", "y-down", "138", "y-down-open",
"39", "y-left", "139", "y-left-open", "40", "y-right", "140", "y-right-open",
"41", "line-ew", "141", "line-ew-open", "42", "line-ns", "142", "line-ns-open",
"43", "line-ne", "143", "line-ne-open", "44", "line-nw", "144", "line-nw-open",
"45", "arrow-up", "145", "arrow-up-open", "46", "arrow-down", "146",
"arrow-down-open", "47", "arrow-left", "147", "arrow-left-open", "48",
"arrow-right", "148", "arrow-right-open", "49", "arrow-bar-up", "149",
"arrow-bar-up-open", "50", "arrow-bar-down", "150", "arrow-bar-down-open",
"51", "arrow-bar-left", "151", "arrow-bar-left-open", "52", "arrow-bar-right",
"152", "arrow-bar-right-open", "53", "arrow", "153", "arrow-open", "54",
"arrow-wide", "154", "arrow-wide-open", 0, 100, 200, 300, 1, 101, 201, 301,
2, 102, 202, 302, 3, 103, 203, 303, 4, 104, 204, 304, 5, 105, 205, 305, 6, 106,
206, 306, 7, 107, 207, 307, 8, 108, 208, 308, 9, 109, 209, 309, 10, 110, 210, 310,
11, 111, 211, 311, 12, 112, 212, 312, 13, 113, 213, 313, 14, 114, 214, 314, 15,
115, 215, 315, 16, 116, 216, 316, 17, 117, 217, 317, 18, 118, 218, 318, 19, 119,

219, 319, 20, 120, 220, 320, 21, 121, 221, 321, 22, 122, 222, 322, 23, 123, 223, 323, 24, 124, 224, 324, 25, 125, 26, 126, 27, 127, 28, 128, 29, 129, 30, 130, 31, 131, 32, 132, 33, 133, 34, 134, 35, 135, 36, 136, 236, 336, 37, 137, 38, 138, 39, 139, 40, 140, 41, 141, 42, 142, 43, 143, 44, 144, 45, 145, 46, 146, 47, 147, 48, 148, 49, 149, 50, 150, 51, 151, 52, 152, 53, 153, 54, 154.

`outlierwidth` Sets the border line width (in px) of the outlier sample points. Number in the interval [0, Inf].

Details

`go_violin()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_violin()` creates an object and adds it to a figure.

Nested objects documented here:

`go_violin_box()` `violin.box`

`go_violin_box_line()` `violin.box.line`

`go_violin_hoverlabel()` `violin.hoverlabel`

`go_violin_hoverlabel_font()` `violin.hoverlabel.font`: Sets the font used in hover labels.

`go_violin_legendgrouptitle()` `violin.legendgrouptitle`

`go_violin_legendgrouptitle_font()` `violin.legendgrouptitle.font`: Sets this legend group's title font.

`go_violin_line()` `violin.line`

`go_violin_marker()` `violin.marker`

`go_violin_marker_line()` `violin.marker.line`

`go_violin_meanline()` `violin.meanline`

`go_violin_selected()` `violin.selected`

`go_violin_selected_marker()` `violin.selected.marker`

`go_violin_unselected()` `violin.unselected`

`go_violin_unselected_marker()` `violin.unselected.marker`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_violin(x = 1:3, y = c(2, 1, 3), marker_color = "red")
```

add_volume *Volume traces*

Description

Draws volume trace between iso-min and iso-max values with coordinates given by four 1-dimensional arrays containing the ‘value‘, ‘x‘, ‘y‘ and ‘z‘ of every vertex of a uniform or non-uniform 3-D grid. Horizontal or vertical slices, caps as well as spaceframe between iso-min and iso-max values could also be drawn using this trace.

Usage

```
add_volume(fig, ...)

## Default S3 method:
add_volume(
  fig,
  ...,
  data = NULL,
  autocolorscale = NULL,
  caps = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  contour = NULL,
  customdata = NULL,
  flatshading = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  isomax = NULL,
  isomin = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lighting = NULL,
  lightposition = NULL,
  meta = NULL,
```

```
  name = NULL,
  opacity = NULL,
  opacityscale = NULL,
  reversescale = NULL,
  scene = NULL,
  showlegend = NULL,
  showscale = NULL,
  slices = NULL,
  spaceframe = NULL,
  surface = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  value = NULL,
  valuehoverformat = NULL,
  visible = NULL,
  x = NULL,
  xhoverformat = NULL,
  y = NULL,
  yhoverformat = NULL,
  z = NULL,
  zhoverformat = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)

go_volume(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  caps = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  contour = NULL,
  customdata = NULL,
  flatshading = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
```

```
  isomax = NULL,
  isomin = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lighting = NULL,
  lightposition = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  opacityscale = NULL,
  reversescale = NULL,
  scene = NULL,
  showlegend = NULL,
  showscale = NULL,
  slices = NULL,
  spaceframe = NULL,
  surface = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  value = NULL,
  valuehoverformat = NULL,
  visible = NULL,
  x = NULL,
  xhoverformat = NULL,
  y = NULL,
  yhoverformat = NULL,
  z = NULL,
  zhoverformat = NULL,
  skip_invalid = FALSE
)

go_volume_caps(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_volume_caps_x(
  arg = NULL,
  ...,
  fill = NULL,
```

```
  show = NULL,
  skip_invalid = FALSE
)

go_volume_caps_y(
  arg = NULL,
  ...,
  fill = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_volume_caps_z(
  arg = NULL,
  ...,
  fill = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_volume_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
```

```
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_volume_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_volume_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
```

```
)

go_volume_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_volume_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_volume_contour(
  arg = NULL,
  ...,
  color = NULL,
  show = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_volume_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_volume_hoverlabel_font(
```

```
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_volume_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_volume_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_volume_lighting(
  arg = NULL,
  ...,
  ambient = NULL,
  diffuse = NULL,
  facenormalsepsilon = NULL,
  fresnel = NULL,
  roughness = NULL,
  specular = NULL,
  vertexnormalsepsilon = NULL,
  skip_invalid = FALSE
```

```
)

go_volume_lightposition(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_volume_slices(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_volume_slices_x(
  arg = NULL,
  ...,
  fill = NULL,
  locations = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_volume_slices_y(
  arg = NULL,
  ...,
  fill = NULL,
  locations = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_volume_slices_z(
  arg = NULL,
  ...,
  fill = NULL,
  locations = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_volume_spaceframe(
```

```
  arg = NULL,
  ...,
  fill = NULL,
  show = NULL,
  skip_invalid = FALSE
)

go_volume_surface(
  arg = NULL,
  ...,
  count = NULL,
  fill = NULL,
  pattern = NULL,
  show = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
autocolorscale	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
caps	Named list or go_volume_caps() object.
cauto	Determines whether or not the color domain is computed with respect to the input data (here ‘value‘) or the bounds set in ‘cmin‘ and ‘cmax‘ Defaults to ‘false‘ when ‘cmin‘ and ‘cmax‘ are set by the user. Logical.
cmax	Sets the upper bound of the color domain. Value should have the same units as ‘value‘ and if set, ‘cmin‘ must be set as well. Number.
cmid	Sets the mid-point of the color domain by scaling ‘cmin‘ and/or ‘cmax‘ to be equidistant to this point. Value should have the same units as ‘value‘. Has no effect when ‘cauto‘ is ‘false‘. Number.
cmin	Sets the lower bound of the color domain. Value should have the same units as ‘value‘ and if set, ‘cmax‘ must be set as well. Number.
coloraxis	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: "coloraxis", "coloraxis2", "coloraxis3", ...

colorbar Named list or go_volume_colorbar() object.

colorscale Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the bounds of the colorscale in color space, use 'cmin' and 'cmax'. Alternatively, 'colorscale' may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

contour Named list or go_volume_contour() object.

customdata Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

flatshading Determines whether or not normal smoothing is applied to the meshes, creating meshes with an angular, low-poly look via flat reflections. Logical.

hoverinfo Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of "x", "y", "z", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.

hoverlabel Named list or go_volume_hoverlabel() object.

hovertemplate Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

hovertemplatefallback Fallback string that's displayed when a variable referenced in a template is miss-

ing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Same as 'text'. String, or a vector of these.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`isomax` Sets the maximum boundary for iso-surface plot. Number.

`isomin` Sets the minimum boundary for iso-surface plot. Number.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
Named list or `go_volume_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`lighting` Named list or `go_volume_lighting()` object.

`lightposition` Named list or `go_volume_lightposition()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

`name` `go_volume()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_volume_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

`opacity` Sets the opacity of the surface. Please note that in the case of using high ‘opacity‘ values for example a value greater than or equal to 0.5 on two surfaces (and 0.25 with four surfaces), an overlay of multiple transparent surfaces may not perfectly be sorted in depth by the webgl API. This behavior may be improved in the near future and is subject to change. Number in the interval [0, 1].

`opacityscale` Sets the opacityscale. The opacityscale must be an array containing arrays mapping a normalized value to an opacity value. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, 1], [0.5, 0.2], [1, 1]]‘ means that higher/lower values would have higher opacity values and those in the middle would be more transparent Alternatively, ‘opacityscale‘ may be a palette name string of the following list: ’min’, ’max’, ’extremes’ and ’uniform’. The default is ’uniform’. Any value.

`reversescale` Reverses the color mapping if true. If true, ‘cmin‘ will correspond to the last color in the array and ‘cmax‘ will correspond to the first color. Logical.

`scene` Sets a reference between this trace’s 3D coordinate system and a 3D scene. If "scene" (the default value), the (x,y,z) coordinates refer to ‘layout.scene‘. If "scene2", the (x,y,z) coordinates refer to ‘layout.scene2‘, and so on. Subplot id: `"scene"`, `"scene2"`, `"scene3"`, ...

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Logical.

`slices` Named list or `go_volume_slices()` object.

`spaceframe` Named list or `go_volume_spaceframe()` object.

`surface` Named list or `go_volume_surface()` object.

`text` `go_volume()`: Sets the text elements associated with the vertices. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_volume_colorbar_title()`: Sets the title of the color bar. String.

`go_volume_legendgrouptitle()`: Sets the title of the legend group. String.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`value` `go_volume()`: Sets the 4th dimension (value) of the vertices. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_volume_colorbar_tickformatstop()`: string - dtickformat for described zoom level, the same as "tickformat" String.

valuehoverformat
: Sets the hover text formatting rule for ‘value‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

visible
: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

x
: `go_volume()`: Sets the X coordinates of the vertices on X axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_volume_caps()`: Named list or `go_volume_caps_x()` object.

`go_volume_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`go_volume_lightposition()`: Numeric vector, representing the X coordinate for each vertex. Number in the interval [-100000, 100000].

`go_volume_slices()`: Named list or `go_volume_slices_x()` object.

xhoverformat
: Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

y
: `go_volume()`: Sets the Y coordinates of the vertices on Y axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_volume_caps()`: Named list or `go_volume_caps_y()` object.

`go_volume_colorbar()`: Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

`go_volume_lightposition()`: Numeric vector, representing the Y coordinate for each vertex. Number in the interval [-100000, 100000].

`go_volume_slices()`: Named list or `go_volume_slices_y()` object.

yhoverformat
: Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

z
: go_volume(): Sets the Z coordinates of the vertices on Z axis. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_volume_caps(): Named list or go_volume_caps_z() object.

go_volume_lightposition(): Numeric vector, representing the Z coordinate for each vertex. Number in the interval [-100000, 100000].

go_volume_slices(): Named list or go_volume_slices_z() object.

zhoverformat
: Sets the hover text formatting rule for ‘z‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘zaxis.hoverformat‘. String.

row
: Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.

col
: Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

skip_invalid
: If TRUE, silently drop invalid properties instead of raising an error.

arg
: Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

fill
: go_volume_caps_x(), go_volume_caps_y(), go_volume_caps_z(): Sets the fill ratio of the ‘caps‘. The default fill value of the ‘caps‘ is 1 meaning that they are entirely shaded. On the other hand Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Number in the interval [0, 1].

go_volume_slices_x(), go_volume_slices_y(), go_volume_slices_z(): Sets the fill ratio of the ‘slices‘. The default fill value of the ‘slices‘ is 1 meaning that they are entirely shaded. On the other hand Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Number in the interval [0, 1].

go_volume_spaceframe(): Sets the fill ratio of the ‘spaceframe‘ elements. The default fill value is 1 meaning that they are entirely shaded. Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Number in the interval [0, 1].

go_volume_surface(): Sets the fill ratio of the iso-surface. The default fill value of the surface is 1 meaning that they are entirely shaded. On the other hand Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Number in the interval [0, 1].

show
: go_volume_caps_x(): Sets the fill ratio of the ‘slices‘. The default fill value of the x ‘slices‘ is 1 meaning that they are entirely shaded. On the other hand Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Logical.

go_volume_caps_y(): Sets the fill ratio of the ‘slices‘. The default fill value of the y ‘slices‘ is 1 meaning that they are entirely shaded. On the other hand

Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Logical.

`go_volume_caps_z()`: Sets the fill ratio of the ‘slices‘. The default fill value of the z ‘slices‘ is 1 meaning that they are entirely shaded. On the other hand Applying a ‘fill‘ ratio less than one would allow the creation of openings parallel to the edges. Logical.

`go_volume_contour()`: Sets whether or not dynamic contours are shown on hover Logical.

`go_volume_slices_x()`: Determines whether or not slice planes about the x dimension are drawn. Logical.

`go_volume_slices_y()`: Determines whether or not slice planes about the y dimension are drawn. Logical.

`go_volume_slices_z()`: Determines whether or not slice planes about the z dimension are drawn. Logical.

`go_volume_spaceframe()`: Displays/hides tetrahedron shapes between minimum and maximum iso-values. Often useful when either caps or surfaces are disabled or filled with values less than 1. Logical.

`go_volume_surface()`: Hides/displays surfaces between minimum and maximum iso-values. Logical.

`bgcolor` `go_volume_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

`go_volume_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`bordercolor` `go_volume_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

`go_volume_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is

wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar's tick label font Named list or go_volume_colorbar_tickfont() object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or go_volume_colorbar_tickformatstop() objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelstep Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

ticklen Sets the tick length (in px). Number in the interval [0, Inf].

tickmode Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of "auto", "linear", "array".

tickprefix Sets a tick label prefix. String.

ticks Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of "outside", "inside", "".

ticksuffix Sets a tick label suffix. String.

ticktext Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals — Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth — Sets the tick width (in px). Number in the interval [0, Inf].

title — String (the title text), named list or go_volume_colorbar_title() object.

xanchor — Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of "left", "center", "right".

xpad — Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".

yanchor — Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

color — go_volume_colorbar_tickfont(), go_volume_colorbar_title_font(), go_volume_legendgroupt
Color: a CSS color string or R color name.

go_volume_contour(): Sets the color of the contour lines. Color: a CSS color string or R color name.

go_volume_hoverlabel_font(): Color: a CSS color string or R color name, or a vector of these.

family — go_volume_colorbar_tickfont(), go_volume_colorbar_title_font(), go_volume_legendgroupt
HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String.

go_volume_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String, or a vector of these.

lineposition — go_volume_colorbar_tickfont(), go_volume_colorbar_title_font(), go_volume_legendgroupt
Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

go_volume_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

shadow
go_volume_colorbar_tickfont(), go_volume_colorbar_title_font(), go_volume_legendgroupt
Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

go_volume_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size
go_volume_colorbar_tickfont(), go_volume_colorbar_title_font(), go_volume_legendgroupt
Number in the interval [1, Inf].

go_volume_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

style
go_volume_colorbar_tickfont(), go_volume_colorbar_title_font(), go_volume_legendgroupt
Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

go_volume_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

textcase
go_volume_colorbar_tickfont(), go_volume_colorbar_title_font(), go_volume_legendgroupt
Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

go_volume_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant
go_volume_colorbar_tickfont(), go_volume_colorbar_title_font(), go_volume_legendgroupt
Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

go_volume_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight
go_volume_colorbar_tickfont(), go_volume_colorbar_title_font(), go_volume_legendgroupt
Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

go_volume_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange
range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled
Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname
: Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

font
: `go_volume_colorbar_title()`: Sets this color bar's title font. Named list or `go_volume_colorbar_title_font()` object.

  `go_volume_hoverlabel()`: Sets the font used in hover labels. Named list or `go_volume_hoverlabel_font()` object.

  `go_volume_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_volume_legendgrouptitle_font()` object.

side
: Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.

width
: Sets the width of the contour lines. Number in the interval [1, 16].

align
: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to ‘namelength - 3‘ characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

ambient
: Ambient light increases overall color visibility but can wash out the image. Number in the interval [0, 1].

diffuse
: Represents the extent that incident rays are reflected in a range of angles. Number in the interval [0, 1].

facenormalsepsilon
: Epsilon for face normals calculation avoids math issues arising from degenerate geometry. Number in the interval [0, 1].

fresnel
: Represents the reflectance as a dependency of the viewing angle; e.g. paper is reflective when viewing it from the edge of the paper (almost 90 degrees), causing shine. Number in the interval [0, 5].

roughness
: Alters specular reflection; the rougher the surface, the wider and less contrasty the shine. Number in the interval [0, 1].

specular
: Represents the level that incident rays are reflected in a single direction, causing shine. Number in the interval [0, 2].

vertexnormalsepsilon
: Epsilon for vertex normals calculation avoids math issues arising from degenerate geometry. Number in the interval [0, 1].

`locations` `go_volume_slices_x()`: Specifies the location(s) of slices on the axis. When not specified slices would be created for all points of the axis x except start and end. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_volume_slices_y()`: Specifies the location(s) of slices on the axis. When not specified slices would be created for all points of the axis y except start and end. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_volume_slices_z()`: Specifies the location(s) of slices on the axis. When not specified slices would be created for all points of the axis z except start and end. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`count` Sets the number of iso-surfaces between minimum and maximum iso-values. By default this value is 2 meaning that only minimum and maximum surfaces would be drawn. Integer in the interval [1, Inf].

`pattern` Sets the surface pattern of the iso-surface 3-D sections. The default pattern of the surface is 'all' meaning that the rest of surface elements would be shaded. The check options (either 1 or 2) could be used to draw half of the squares on the surface. Using various combinations of capital 'A', 'B', 'C', 'D' and 'E' may also be used to reduce the number of triangles on the iso-surfaces and creating other patterns of interest. Any combination of `"A"`, `"B"`, `"C"`, `"D"`, `"E"` joined with `"+"`, or one of `"all"`, `"odd"`, `"even"`.

Details

`go_volume()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_volume()` creates an object and adds it to a figure.

Nested objects documented here:

`go_volume_caps()` `volume.caps`

`go_volume_caps_x()` `volume.caps.x`

`go_volume_caps_y()` `volume.caps.y`

`go_volume_caps_z()` `volume.caps.z`

`go_volume_colorbar()` `volume.colorbar`

`go_volume_colorbar_tickfont()` `volume.colorbar.tickfont`: Sets the color bar's tick label font

`go_volume_colorbar_tickformatstop()` `volume.colorbar.tickformatstop`

`go_volume_colorbar_title()` `volume.colorbar.title`

`go_volume_colorbar_title_font()` `volume.colorbar.title.font`: Sets this color bar's title font.

`go_volume_contour()` `volume.contour`

`go_volume_hoverlabel()` `volume.hoverlabel`

`go_volume_hoverlabel_font()` `volume.hoverlabel.font`: Sets the font used in hover labels.

`go_volume_legendgrouptitle()` `volume.legendgrouptitle`

`go_volume_legendgrouptitle_font()` `volume.legendgrouptitle.font`: Sets this legend group's title font.

go_volume_lighting() volume.lighting

go_volume_lightposition() volume.lightposition

go_volume_slices() volume.slices

go_volume_slices_x() volume.slices.x

go_volume_slices_y() volume.slices.y

go_volume_slices_z() volume.slices.z

go_volume_spaceframe() volume.spaceframe

go_volume_surface() volume.surface

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_volume(x = 1:3, y = c(2, 1, 3))
```

---

add_waterfall *Waterfall traces*

Description

Draws waterfall trace which is useful graph to displays the contribution of various elements (either positive or negative) in a bar chart. The data visualized by the span of the bars is set in ‘y‘ if ‘orientation‘ is set to "v" (the default) and the labels are set in ‘x‘. By setting ‘orientation‘ to "h", the roles are interchanged.

Usage

```
add_waterfall(fig, ...)

## Default S3 method:
add_waterfall(
  fig,
  ...,
  data = NULL,
  alignmentgroup = NULL,
  base = NULL,
  cliponaxis = NULL,
  connector = NULL,
```

```
    constraintext = NULL,
    customdata = NULL,
    decreasing = NULL,
    dx = NULL,
    dy = NULL,
    hoverinfo = NULL,
    hoverlabel = NULL,
    hovertemplate = NULL,
    hovertemplatefallback = NULL,
    hovertext = NULL,
    ids = NULL,
    increasing = NULL,
    insidetextanchor = NULL,
    insidetextfont = NULL,
    legend = NULL,
    legendgroup = NULL,
    legendgrouptitle = NULL,
    legendrank = NULL,
    legendwidth = NULL,
    measure = NULL,
    meta = NULL,
    name = NULL,
    offset = NULL,
    offsetgroup = NULL,
    opacity = NULL,
    orientation = NULL,
    outsidetextfont = NULL,
    selectedpoints = NULL,
    showlegend = NULL,
    text = NULL,
    textangle = NULL,
    textfont = NULL,
    textinfo = NULL,
    textposition = NULL,
    texttemplate = NULL,
    texttemplatefallback = NULL,
    totals = NULL,
    uid = NULL,
    uirevision = NULL,
    visible = NULL,
    width = NULL,
    x = NULL,
    x0 = NULL,
    xaxis = NULL,
    xhoverformat = NULL,
    xperiod = NULL,
    xperiod0 = NULL,
    xperiodalignment = NULL,
```

```
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  yperiod = NULL,
  yperiod0 = NULL,
  yperiodalignment = NULL,
  zorder = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)

go_waterfall(
  arg = NULL,
  ...,
  alignmentgroup = NULL,
  base = NULL,
  cliponaxis = NULL,
  connector = NULL,
  constraintext = NULL,
  customdata = NULL,
  decreasing = NULL,
  dx = NULL,
  dy = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  increasing = NULL,
  insidetextanchor = NULL,
  insidetextfont = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  measure = NULL,
  meta = NULL,
  name = NULL,
  offset = NULL,
  offsetgroup = NULL,
  opacity = NULL,
  orientation = NULL,
  outsidetextfont = NULL,
```

```
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textangle = NULL,
  textfont = NULL,
  textinfo = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  totals = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  width = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  yperiod = NULL,
  yperiod0 = NULL,
  yperiodalignment = NULL,
  zorder = NULL,
  skip_invalid = FALSE
)

go_waterfall_connector(
  arg = NULL,
  ...,
  line = NULL,
  mode = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_waterfall_connector_line(
  arg = NULL,
  ...,
  color = NULL,
  dash = NULL,
  width = NULL,
  skip_invalid = FALSE
```

```
)

go_waterfall_decreasing(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

go_waterfall_decreasing_marker(
  arg = NULL,
  ...,
  color = NULL,
  line = NULL,
  skip_invalid = FALSE
)

go_waterfall_decreasing_marker_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_waterfall_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_waterfall_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_waterfall_increasing(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)
```

```
  go_waterfall_increasing_marker(
    arg = NULL,
    ...,
    color = NULL,
    line = NULL,
    skip_invalid = FALSE
  )

  go_waterfall_increasing_marker_line(
    arg = NULL,
    ...,
    color = NULL,
    width = NULL,
    skip_invalid = FALSE
  )

  go_waterfall_insidetextfont(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_waterfall_legendgrouptitle(
    arg = NULL,
    ...,
    font = NULL,
    text = NULL,
    skip_invalid = FALSE
  )

  go_waterfall_legendgrouptitle_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
```

```
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_waterfall_outsidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_waterfall_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_waterfall_totals(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

go_waterfall_totals_marker(
  arg = NULL,
  ...,
  color = NULL,
  line = NULL,
  skip_invalid = FALSE
)

go_waterfall_totals_marker_line(
```

```
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)
```

Arguments

fig	A go_figure object.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
alignmentgroup	Set several traces linked to the same position axis or matching axes to the same alignmentgroup. This controls whether bars compute their positional range dependently or independently. String.
base	Sets where the bar base is drawn (in position axis units). Number.
cliponaxis	Determines whether the text nodes are clipped about the subplot axes. To show the text nodes above axis lines and tick labels, make sure to set 'xaxis.layer' and 'yaxis.layer' to "below traces". Logical.
connector	Named list or go_waterfall_connector() object.
constraintext	Constrain the size of text inside or outside a bar to be no larger than the bar itself. One of "inside", "outside", "both", "none".
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
decreasing	Named list or go_waterfall_decreasing() object.
dx	Sets the x coordinate step. See 'x0' for more info. Number.
dy	Sets the y coordinate step. See 'y0' for more info. Number.
hoverinfo	Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of "name", "x", "y", "text", "initial", "delta", "final" joined with "+", or one of "all", "none", "skip", or a vector of these.
hoverlabel	Named list or go_waterfall_hoverlabel() object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to

those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'initial', 'delta' and 'final'. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`increasing`
: Named list or `go_waterfall_increasing()` object.

`insidetextanchor`
: Determines if texts are kept at center or start/end points in 'textposition' "inside" mode. One of `"end"`, `"middle"`, `"start"`.

`insidetextfont`
: Sets the font used for 'text' lying inside the bar. Named list or `go_waterfall_insidetextfont()` object.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_waterfall_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less

than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`measure` An array containing types of values. By default the values are considered as 'relative'. However; it is possible to use 'total' to compute the sums. Also 'absolute' could be applied to reset the computed total or to declare an initial value where needed. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

`name` Sets the trace name. The trace name appears as the legend item and on hover. String.

`offset` Shifts the position where the bar is drawn (in position axis units). In "group" barmode, traces that set "offset" will be excluded and drawn in "overlay" mode instead. Number, or a vector of these.

`offsetgroup` Set several traces linked to the same position axis or matching axes to the same offsetgroup where bars of the same position coordinate will line up. String.

`opacity` Sets the opacity of the trace. Number in the interval [0, 1].

`orientation` Sets the orientation of the bars. With "v" ("h"), the value of the each bar spans along the vertical (horizontal). One of `"v"`, `"h"`.

`outsidetextfont`

Sets the font used for 'text' lying outside the bar. Named list or `go_waterfall_outsidetextfont()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the 'unselected' are turned on for all points, whereas, any other non-array values means no selection all where the 'selected' and 'unselected' styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`text` `go_waterfall()`: Sets text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. If trace 'hoverinfo' contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_waterfall_legendgrouptitle()`: Sets the title of the legend group. String.

textangle — Sets the angle of the tick labels with respect to the bar. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. With "auto" the texts may automatically be rotated to fit with the maximum size in bars. Angle in degrees.

textfont — Sets the font used for ‘text‘. Named list or go_waterfall_textfont() object.

textinfo — Determines what trace information appears on the graph. In the case of having multiple waterfalls, totals are computed separately (per trace). Flags are rendered in a fixed order; use ‘texttemplate‘ if you need explicit control over the rendered string, including the order of fields and surrounding text. Any combination of "label", "text", "initial", "delta", "final" joined with "+", or one of "none".

textposition — Specifies the location of the ‘text‘. "inside" positions ‘text‘ inside, next to the bar end (rotated and scaled if needed). "outside" positions ‘text‘ outside, next to the bar end (scaled if needed), unless there is another bar stacked on this one, then the text gets pushed inside. "auto" tries to position ‘text‘ inside the bar, but if the bar is too small and no bar is stacked on this one the text is moved outside. If "none", no text appears. One of "inside", "outside", "auto", "none", or a vector of these.

texttemplate — Template string used for rendering the information text that appears on points. Note that this will override ‘textinfo‘. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variables ‘initial‘, ‘delta‘, ‘final‘ and ‘label‘. String, or a vector of these.

texttemplatefallback — Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

totals — Named list or go_waterfall_totals() object.

uid — Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision — Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can

add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`visible` `go_waterfall()`: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`go_waterfall_connector()`: Determines if connector lines are drawn. Logical.

`width` `go_waterfall()`: Sets the bar width (in position axis units). Number in the interval [0, Inf], or a vector of these.

`go_waterfall_connector_line()`: Sets the line width (in px). Number in the interval [0, Inf].

`go_waterfall_decreasing_marker_line()`: Sets the line width of all decreasing values. Number in the interval [0, Inf].

`go_waterfall_increasing_marker_line()`: Sets the line width of all increasing values. Number in the interval [0, Inf].

`go_waterfall_totals_marker_line()`: Sets the line width of all intermediate sums and total values. Number in the interval [0, Inf].

`x` Sets the x coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`x0` Alternate to ‘x‘. Builds a linear space of x coordinates. Use with ‘dx‘ where ‘x0‘ is the starting coordinate and ‘dx‘ the step. Any value.

`xaxis` Sets a reference between this trace’s x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

`xhoverformat` Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

`xperiod` Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the x axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

`xperiod0` Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the x0 axis. When ‘x0period‘ is round number of weeks, the ‘x0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

`xperiodalignment`
Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the x axis. One of `"start"`, `"middle"`, `"end"`.

`y` Sets the y coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`y0` Alternate to ‘y‘. Builds a linear space of y coordinates. Use with ‘dy‘ where ‘y0‘ is the starting coordinate and ‘dy‘ the step. Any value.

`yaxis` Sets a reference between this trace’s y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...

`yhoverformat` Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

`yperiod` Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the y axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

`yperiod0` Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the y0 axis. When ‘y0period‘ is round number of weeks, the ‘y0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

`yperiodalignment` Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the y axis. One of `"start"`, `"middle"`, `"end"`.

`zorder` Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher ‘zorder‘ appear in front of those with lower ‘zorder‘. Integer.

`row` Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`col` Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

`secondary_y` If `TRUE`, use the secondary y axis of the subplot.

`skip_invalid` If `TRUE`, silently drop invalid properties instead of raising an error.

`arg` Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

`line` `go_waterfall_connector()`: Named list or `go_waterfall_connector_line()` object.

`go_waterfall_decreasing_marker()`: Named list or `go_waterfall_decreasing_marker_line()` object.

`go_waterfall_increasing_marker()`: Named list or `go_waterfall_increasing_marker_line()` object.

`go_waterfall_totals_marker()`: Named list or `go_waterfall_totals_marker_line()` object.

`mode` Sets the shape of connector lines. One of `"spanning"`, `"between"`.

color
: go_waterfall_connector_line(): Sets the line color. Color: a CSS color string or R color name.
go_waterfall_decreasing_marker(): Sets the marker color of all decreasing values. Color: a CSS color string or R color name.
go_waterfall_decreasing_marker_line(): Sets the line color of all decreasing values. Color: a CSS color string or R color name.
go_waterfall_hoverlabel_font(),go_waterfall_insidetextfont(),go_waterfall_outsidetex
go_waterfall_textfont(): Color: a CSS color string or R color name, or a vector of these.
go_waterfall_increasing_marker(): Sets the marker color of all increasing values. Color: a CSS color string or R color name.
go_waterfall_increasing_marker_line(): Sets the line color of all increasing values. Color: a CSS color string or R color name.
go_waterfall_legendgrouptitle_font(): Color: a CSS color string or R color name.
go_waterfall_totals_marker(): Sets the marker color of all intermediate sums and total values. Color: a CSS color string or R color name.
go_waterfall_totals_marker_line(): Sets the line color of all intermediate sums and total values. Color: a CSS color string or R color name.

dash
: Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$".

marker
: go_waterfall_decreasing(): Named list or go_waterfall_decreasing_marker() object.
go_waterfall_increasing(): Named list or go_waterfall_increasing_marker() object.
go_waterfall_totals(): Named list or go_waterfall_totals_marker() object.

align
: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.

bgcolor
: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

bordercolor
: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

font
: go_waterfall_hoverlabel(): Sets the font used in hover labels. Named list or go_waterfall_hoverlabel_font() object.
go_waterfall_legendgrouptitle(): Sets this legend group's title font. Named list or go_waterfall_legendgrouptitle_font() object.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

family
: go_waterfall_hoverlabel_font(), go_waterfall_insidetextfont(), go_waterfall_outsidetex go_waterfall_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

  go_waterfall_legendgrouptitle_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: go_waterfall_hoverlabel_font(), go_waterfall_insidetextfont(), go_waterfall_outsidetex go_waterfall_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

  go_waterfall_legendgrouptitle_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow
: go_waterfall_hoverlabel_font(), go_waterfall_insidetextfont(), go_waterfall_outsidetex go_waterfall_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

  go_waterfall_legendgrouptitle_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: go_waterfall_hoverlabel_font(), go_waterfall_insidetextfont(), go_waterfall_outsidetex go_waterfall_textfont(): Number in the interval [1, Inf], or a vector of these.

  go_waterfall_legendgrouptitle_font(): Number in the interval [1, Inf].

style
: go_waterfall_hoverlabel_font(), go_waterfall_insidetextfont(), go_waterfall_outsidetex go_waterfall_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

  go_waterfall_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_waterfall_hoverlabel_font(), go_waterfall_insidetextfont(), go_waterfall_outsidetex go_waterfall_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_waterfall_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant — go_waterfall_hoverlabel_font(), go_waterfall_insidetextfont(), go_waterfall_outsidetex
go_waterfall_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_waterfall_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight — go_waterfall_hoverlabel_font(), go_waterfall_insidetextfont(), go_waterfall_outsidetex
go_waterfall_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_waterfall_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

Details

go_waterfall() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_waterfall() creates an object and adds it to a figure.

Nested objects documented here:

go_waterfall_connector() waterfall.connector

go_waterfall_connector_line() waterfall.connector.line

go_waterfall_decreasing() waterfall.decreasing

go_waterfall_decreasing_marker() waterfall.decreasing.marker

go_waterfall_decreasing_marker_line() waterfall.decreasing.marker.line

go_waterfall_hoverlabel() waterfall.hoverlabel

go_waterfall_hoverlabel_font() waterfall.hoverlabel.font: Sets the font used in hover labels.

go_waterfall_increasing() waterfall.increasing

go_waterfall_increasing_marker() waterfall.increasing.marker

go_waterfall_increasing_marker_line() waterfall.increasing.marker.line

go_waterfall_insidetextfont() waterfall.insidetextfont: Sets the font used for 'text' lying inside the bar.

go_waterfall_legendgrouptitle() waterfall.legendgrouptitle

go_waterfall_legendgrouptitle_font() waterfall.legendgrouptitle.font: Sets this legend group's title font.

go_waterfall_outsidetextfont() waterfall.outsidetextfont: Sets the font used for 'text' lying outside the bar.

go_waterfall_textfont() waterfall.textfont: Sets the font used for 'text'.

go_waterfall_totals() waterfall.totals

go_waterfall_totals_marker() waterfall.totals.marker

go_waterfall_totals_marker_line() waterfall.totals.marker.line

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_waterfall(x = 1:3, y = c(2, 1, 3))
```

---

as_float32 *Send numeric data to the browser in single precision*

Description

Marks a numeric vector or matrix to be sent to plotly.js as a float32 typed array: 4 bytes per value instead of 8, halving the size of the widget's data and of plotly.js's copy of it. Use it in trace properties, or on data frame columns used by `px_scatter()` and the other `px_*()` functions. Set `options(plotly.core.float32 = TRUE)` to send all numeric arrays in single precision.

Usage

```
as_float32(x)
```

Arguments

x	A numeric vector or matrix.

Details

Single precision keeps about 7 significant digits: values shown in hover labels and returned by `event_data()` change (`0.1` becomes `0.100000001490116`), whole numbers above 2^24 (16,777,216) aren't exact and numeric times in seconds since 1970 are only accurate to minutes. Vectors of whole numbers (in the 32-bit integer range) are always sent as integers without loss, and date-times as text, so neither is affected. WebGL traces (e.g. `go_scattergl()`) draw in single precision anyway.

Value

`x`, marked with class `plotly_float32`.

Examples

```
n <- 1e4
data.frame(x = rnorm(n), y = rnorm(n)) |>
  go_figure() |>
  add_scattergl(x = ~as_float32(x), y = ~as_float32(y), mode = "markers")
```

`chrome_exporter` *Save a figure as a static image*

Description

Exports a figure to PNG, JPEG, WebP, SVG, PDF or EPS with plotly.js's own image export, using the plotly.js bundled with this package in headless Chrome (driven by the chromote package). No Python is needed.

Usage

```
chrome_exporter(...)

save_image(p, file, ..., width = NULL, height = NULL, scale = NULL)
```

Arguments

`...`	Unused.
`p`	A `go_figure()`, a figure from a `px_*()` function or a plotly htmlwidget (anything `as_widget()` accepts).
`file`	Output file; the format is taken from the extension (`.png`, `.jpeg`, `.jpg`, `.webp`, `.svg`, `.pdf` or `.eps`).
`width, height`	Image size in pixels. Default: the figure's layout width and height, or 700 x 500.
`scale`	Multiplier for the image resolution (e.g. 2 for high-DPI PNGs).

Details

It uses the browser set by the `CHROMOTE_CHROME` environment variable, an installed Google Chrome, another installed Chromium-based browser (Microsoft Edge, which every Windows 10/11 machine has, Chromium or Brave, also when installed for the current user only), or a Chrome Headless Shell downloaded by chromote. Without any of these, an interactive session asks whether to download Chrome Headless Shell (about 100 MB, kept for later sessions); elsewhere, download it once with `chromote::chrome_versions_add("latest-stable", "chrome-headless-shell")`.

The browser is started on the first export and reused for later ones. Large figures can take a while to draw: an export waits up to `getOption("plotly.core.export_timeout", 300)` seconds. LaTeX (see `TeX()`) is typeset with the MathJax bundled with the package, so it works offline. PDF is made by printing the SVG image to a page of the figure's size; EPS is converted from the SVG image with the rsvg package.

Value

`save_image()` returns `file` invisibly. `chrome_exporter()` returns an exporter with `transform()` and `shutdown()` functions.

Downloads

save_image() downloads nothing on its own in scripts, tests or examples. In an interactive session that needs set-up, it asks first (see install_chrome()). What it may download keeps its own licence (see the package's COPYRIGHTS file): Chrome Headless Shell (from Google's Chrome for Testing, kept in chromote's cache), on Linux the shared libraries Chrome needs (your distribution's packages, kept in tools::R_user_dir("plotly.core", "cache")). remove_chrome() removes the downloads.

chrome_exporter() returns an exporter of its own, whose transform(p, file, ...) function writes images; call its shutdown() function when done.

Examples

```
## Not run:
fig <- px_data_iris() |> px_scatter(x = ~sepal_width, y = ~sepal_length, color = ~species)
save_image(fig, tempfile(fileext = ".png"), width = 800, height = 500, scale = 2)
save_image(fig, tempfile(fileext = ".svg"))

## End(Not run)
```

---

config *Set plotly.js configuration options*

Description

Sets options of the plotly.js config object (e.g. displayModeBar, scrollZoom, toImageButtonOptions), which control interaction rather than appearance. Options are merged with any set previously. config() is an S3 generic, so packages can handle options of their own plot classes before the default method validates them.

Usage

```
config(p, ...)

## Default S3 method:
config(
  p,
  ...,
  locale = NULL,
  mathjax = NULL,
  cloud = FALSE,
  showSendToCloud = cloud
)
```

Arguments

p	A go_figure() or plotly htmlwidget.
...	Named configuration options; see https://plotly.com/javascript/configuration-options/.
locale	A locale such as "de" or "fr-ch", for translated modebar text and number/date formatting.
mathjax	Load MathJax so LaTeX in $...$ text (see TeX()) is rendered: "local" (or TRUE) for the MathJax 3 bundled with the package, which works offline and is embedded in self-contained HTML (about 2 MB), or "cdn" to load it from cdn.jsdelivr.net.
cloud, showSendToCloud	Show the modebar button that sends the chart to Plotly Cloud (plotlyServerURL)? Off by default, because clicking it uploads the chart data to an external server. cloud is the legacy plotly package's name for it.

Value

The modified figure.

Examples

```
go_figure(go_scatter(y = 1:3)) |>
  config(displayModeBar = FALSE, scrollZoom = TRUE, locale = "de")
```

---

event_data *Access plotly events in Shiny*

Description

Returns the data of the most recent plotly event of a given type, from the plot with the given source (see event_source()); NULL before any such event. Must be called in a reactive context of a Shiny app.

Usage

```
event_data(
 event = c("plotly_hover", "plotly_unhover", "plotly_click", "plotly_doubleclick",
    "plotly_selected", "plotly_selecting", "plotly_brushed", "plotly_brushing",
   "plotly_deselect", "plotly_relayout", "plotly_restyle", "plotly_legendclick",
    "plotly_legenddoubleclick", "plotly_clickannotation", "plotly_afterplot",
    "plotly_sunburstclick"),
  source = "A",
  session = shiny::getDefaultReactiveDomain(),
  priority = c("input", "event")
)
```

Arguments

event	The event type.
source	The source id of the plot.
session	The Shiny session.
priority	"input" (default) triggers reactivity only when the value changes; "event" on every event, even with the same value.

Details

Point events (click, hover, selected, ...) return a data frame with curveNumber, pointNumber, x, y and, where available, z, customdata and key. plotly_relayout returns the changed layout properties (e.g. axis ranges after zooming).

Value

Event data, or NULL.

Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  ui <- shiny::fluidPage(plotlyOutput("plot"), shiny::verbatimTextOutput("click"))
  server <- function(input, output, session) {
   output$plot <- renderPlotly(px_data_iris() |> px_scatter(x = ~sepal_width, y = ~sepal_length))
    output$click <- shiny::renderPrint(event_data("plotly_click"))
  }
  shiny::shinyApp(ui, server)
}
```

---

event_register *Register plotly events for Shiny*

Description

Plots send these events to Shiny by default: plotly_hover, plotly_click, plotly_selected, plotly_relayout, plotly_brushed, plotly_brushing, plotly_clickannotation, plotly_doubleclick, plotly_deselect, plotly_afterplot and plotly_sunburstclick. event_register() adds others (e.g. plotly_legendclick); event_unregister() removes events. event_source() sets the source id that event_data() uses to tell plots apart (default "A").

Usage

```
event_register(p, event = NULL)

event_unregister(p, event = NULL)

event_source(p, source)
```

Arguments

p	A go_figure() or plotly htmlwidget.
event	An event name (see event_data()).
source	A string identifying the plot.

Value

The modified figure.

Examples

```
fig <- go_figure(go_scatter(y = 1:3)) |>
  event_source("scatter") |>
  event_register("plotly_legendclick")
```

---

get_template *Figure templates*

Description

Templates (themes) set default layout and trace properties for figures. The built-in templates are those of plotly.py: "plotly" (the default), "plotly_white", "plotly_dark", "ggplot2", "seaborn", "simple_white", "presentation", "xgridoff", "ygridoff", "gridon" and "none". Templates can be combined with "+", e.g. "plotly_white+presentation"; later templates take precedence.

Usage

```
get_template(name)

template_names()

register_template(name, template)

default_template(name)
```

Arguments

name	Template name, or several joined with "+".
template	A named list with layout and/or data elements, e.g. list(layout = list(font = list(family = "Georgia"))).

Details

Figures without a layout$template use the default template when they are rendered. Plotly Express functions add their template to the figure.

Value

`get_template()` returns the template as a named list. `template_names()` returns the available names. `default_template()` returns the default template name; with an argument it sets the default and returns the previous one invisibly. `register_template()` returns `name` invisibly.

Examples

```
template_names()
tpl <- get_template("plotly_white+presentation")
tpl$layout$font

register_template("big_font", list(layout = list(font = list(size = 20))))
old <- default_template("plotly+big_font")
default_template(old)
```

---

go_bar *Bar traces*

Description

The data visualized by the span of the bars is set in ‘y‘ if ‘orientation‘ is set to "v" (the default) and the labels are set in ‘x‘. By setting ‘orientation‘ to "h", the roles are interchanged.

Usage

```
go_bar(
  arg = NULL,
  ...,
  alignmentgroup = NULL,
  base = NULL,
  cliponaxis = NULL,
  constraintext = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  error_x = NULL,
  error_y = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  insidetextanchor = NULL,
  insidetextfont = NULL,
  legend = NULL,
  legendgroup = NULL,
```

```
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  offset = NULL,
  offsetgroup = NULL,
  opacity = NULL,
  orientation = NULL,
  outsidetextfont = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  textangle = NULL,
  textfont = NULL,
  textposition = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  width = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  yperiod = NULL,
  yperiod0 = NULL,
  yperiodalignment = NULL,
  zorder = NULL,
  skip_invalid = FALSE
)

go_bar_error_x(
  arg = NULL,
  ...,
```

```
  array = NULL,
  arrayminus = NULL,
  color = NULL,
  copy_ystyle = NULL,
  symmetric = NULL,
  thickness = NULL,
  traceref = NULL,
  tracerefminus = NULL,
  type = NULL,
  value = NULL,
  valueminus = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_bar_error_y(
  arg = NULL,
  ...,
  array = NULL,
  arrayminus = NULL,
  color = NULL,
  symmetric = NULL,
  thickness = NULL,
  traceref = NULL,
  tracerefminus = NULL,
  type = NULL,
  value = NULL,
  valueminus = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_bar_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_bar_hoverlabel_font(
  arg = NULL,
```

```
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_bar_insidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_bar_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_bar_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
```

```
  skip_invalid = FALSE
)

go_bar_marker(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  cornerradius = NULL,
  line = NULL,
  opacity = NULL,
  pattern = NULL,
  reversescale = NULL,
  showscale = NULL,
  skip_invalid = FALSE
)

go_bar_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
```

```
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_bar_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_bar_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
```

```
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_bar_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_bar_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_bar_marker_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_bar_marker_pattern(
  arg = NULL,
```

```
  ...,
  bgcolor = NULL,
  fgcolor = NULL,
  fgopacity = NULL,
  fillmode = NULL,
  path = NULL,
  shape = NULL,
  size = NULL,
  solidity = NULL,
  skip_invalid = FALSE
)

go_bar_outsidetextfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_bar_selected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_bar_selected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  skip_invalid = FALSE
)

go_bar_selected_textfont(arg = NULL, ..., color = NULL, skip_invalid = FALSE)

go_bar_textfont(
  arg = NULL,
  ...,
```

```
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_bar_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_bar_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  skip_invalid = FALSE
)

go_bar_unselected_textfont(arg = NULL, ..., color = NULL, skip_invalid = FALSE)

add_bar(fig, ...)

## Default S3 method:
add_bar(
  fig,
  ...,
  data = NULL,
  alignmentgroup = NULL,
  base = NULL,
  cliponaxis = NULL,
  constraintext = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  error_x = NULL,
  error_y = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
```

```
    hovertemplate = NULL,
    hovertemplatefallback = NULL,
    hovertext = NULL,
    ids = NULL,
    insidetextanchor = NULL,
    insidetextfont = NULL,
    legend = NULL,
    legendgroup = NULL,
    legendgrouptitle = NULL,
    legendrank = NULL,
    legendwidth = NULL,
    marker = NULL,
    meta = NULL,
    name = NULL,
    offset = NULL,
    offsetgroup = NULL,
    opacity = NULL,
    orientation = NULL,
    outsidetextfont = NULL,
    selected = NULL,
    selectedpoints = NULL,
    showlegend = NULL,
    text = NULL,
    textangle = NULL,
    textfont = NULL,
    textposition = NULL,
    texttemplate = NULL,
    texttemplatefallback = NULL,
    uid = NULL,
    uirevision = NULL,
    unselected = NULL,
    visible = NULL,
    width = NULL,
    x = NULL,
    x0 = NULL,
    xaxis = NULL,
    xcalendar = NULL,
    xhoverformat = NULL,
    xperiod = NULL,
    xperiod0 = NULL,
    xperiodalignment = NULL,
    y = NULL,
    y0 = NULL,
    yaxis = NULL,
    ycalendar = NULL,
    yhoverformat = NULL,
    yperiod = NULL,
    yperiod0 = NULL,
```

```
  yperiodalignment = NULL,
  zorder = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
alignmentgroup	Set several traces linked to the same position axis or matching axes to the same alignmentgroup. This controls whether bars compute their positional range dependently or independently. String.
base	Sets where the bar base is drawn (in position axis units). In "stack" or "relative" barmode, traces that set "base" will be excluded and drawn in "overlay" mode instead. Any value.
cliponaxis	Determines whether the text nodes are clipped about the subplot axes. To show the text nodes above axis lines and tick labels, make sure to set ‘xaxis.layer‘ and ‘yaxis.layer‘ to "below traces". Logical.
constraintext	Constrain the size of text inside or outside a bar to be no larger than the bar itself. One of `"inside"`, `"outside"`, `"both"`, `"none"`.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
dx	Sets the x coordinate step. See ‘x0‘ for more info. Number.
dy	Sets the y coordinate step. See ‘y0‘ for more info. Number.
error_x	Named list or `go_bar_error_x()` object.
error_y	Named list or `go_bar_error_y()` object.
hoverinfo	Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.
hoverlabel	Named list or `go_bar_hoverlabel()` object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown.

Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'value' and 'label'. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Sets hover text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`insidetextanchor`
: Determines if texts are kept at center or start/end points in 'textposition' "inside" mode. One of `"end"`, `"middle"`, `"start"`.

`insidetextfont`
: Sets the font used for 'text' lying inside the bar. Named list or `go_bar_insidetextfont()` object.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_bar_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`marker` `go_bar()`: Named list or `go_bar_marker()` object.

`go_bar_selected()`: Named list or `go_bar_selected_marker()` object.

`go_bar_unselected()`: Named list or `go_bar_unselected_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_bar()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_bar_marker_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`offset` Shifts the position where the bar is drawn (in position axis units). In "group" barmode, traces that set "offset" will be excluded and drawn in "overlay" mode instead. Number, or a vector of these.

`offsetgroup` Set several traces linked to the same position axis or matching axes to the same offsetgroup where bars of the same position coordinate will line up. String.

`opacity` `go_bar()`: Sets the opacity of the trace. Number in the interval [0, 1].

`go_bar_marker()`: Sets the opacity of the bars. Number in the interval [0, 1], or a vector of these.

`go_bar_selected_marker()`: Sets the marker opacity of selected points. Number in the interval [0, 1].

`go_bar_unselected_marker()`: Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

`orientation` `go_bar()`: Sets the orientation of the bars. With "v" ("h"), the value of the each bar spans along the vertical (horizontal). One of `"v"`, `"h"`.

`go_bar_marker_colorbar()`: Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outsidetextfont`

Sets the font used for ‘text‘ lying outside the bar. Named list or `go_bar_outsidetextfont()` object.

`selected` Named list or `go_bar_selected()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.

showlegend
: Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

text
: go_bar(): Sets text elements associated with each (x,y) pair. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. If trace 'hoverinfo' contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

  go_bar_legendgrouptitle(): Sets the title of the legend group. String.

  go_bar_marker_colorbar_title(): Sets the title of the color bar. String.

textangle
: Sets the angle of the tick labels with respect to the bar. For example, a 'tickangle' of -90 draws the tick labels vertically. With "auto" the texts may automatically be rotated to fit with the maximum size in bars. Angle in degrees.

textfont
: go_bar(): Sets the font used for 'text'. Named list or go_bar_textfont() object.

  go_bar_selected(): Named list or go_bar_selected_textfont() object.

  go_bar_unselected(): Named list or go_bar_unselected_textfont() object.

textposition
: Specifies the location of the 'text'. "inside" positions 'text' inside, next to the bar end (rotated and scaled if needed). "outside" positions 'text' outside, next to the bar end (scaled if needed), unless there is another bar stacked on this one, then the text gets pushed inside. "auto" tries to position 'text' inside the bar, but if the bar is too small and no bar is stacked on this one the text is moved outside. If "none", no text appears. One of "inside", "outside", "auto", "none", or a vector of these.

texttemplate
: Template string used for rendering the information text that appears on points. Note that this will override 'textinfo'. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'value' and 'label'. String, or a vector of these.

texttemplatefallback
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

uid
: Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision
: Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as

‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`unselected` Named list or `go_bar_unselected()` object.

`visible` `go_bar()`: Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`go_bar_error_x()`, `go_bar_error_y()`: Determines whether or not this set of error bars is visible. Logical.

`width` `go_bar()`: Sets the bar width (in position axis units). Number in the interval [0, Inf], or a vector of these.

`go_bar_error_x()`, `go_bar_error_y()`: Sets the width (in px) of the cross-bar at both ends of the error bars. Number in the interval [0, Inf].

`go_bar_marker_line()`: Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

`x` `go_bar()`: Sets the x coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_bar_marker_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`x0` Alternate to ‘x‘. Builds a linear space of x coordinates. Use with ‘dx‘ where ‘x0‘ is the starting coordinate and ‘dx‘ the step. Any value.

`xaxis` Sets a reference between this trace’s x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

`xcalendar` Sets the calendar system to use with ‘x‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

`xhoverformat` Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

xperiod Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the x axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

xperiod0 Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the x0 axis. When ‘x0period‘ is round number of weeks, the ‘x0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

xperiodalignment
Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the x axis. One of `"start"`, `"middle"`, `"end"`.

y go_bar(): Sets the y coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_bar_marker_colorbar(): Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

y0 Alternate to ‘y‘. Builds a linear space of y coordinates. Use with ‘dy‘ where ‘y0‘ is the starting coordinate and ‘dy‘ the step. Any value.

yaxis Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...

ycalendar Sets the calendar system to use with ‘y‘ date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

yhoverformat Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

yperiod Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the y axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

yperiod0 Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the y0 axis. When ‘y0period‘ is round number of weeks, the ‘y0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

yperiodalignment
Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the y axis. One of `"start"`, `"middle"`, `"end"`.

zorder	Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher ‘zorder‘ appear in front of those with lower ‘zorder‘. Integer.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
array	Sets the data corresponding the length of each error bar. Values are plotted relative to the underlying data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
arrayminus	Sets the data corresponding the length of each error bar in the bottom (left) direction for vertical (horizontal) bars Values are plotted relative to the underlying data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
color	go_bar_error_x(), go_bar_error_y(): Sets the stroke color of the error bars. Color: a CSS color string or R color name. go_bar_hoverlabel_font(), go_bar_insidetextfont(), go_bar_outsidetextfont(), go_bar_textfont(): Color: a CSS color string or R color name, or a vector of these. go_bar_legendgrouptitle_font(), go_bar_marker_colorbar_tickfont(), go_bar_marker_colorbar_title_font(): Color: a CSS color string or R color name. go_bar_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.cmin‘ and ‘marker.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by bar.marker.colorscale, or a vector of these. go_bar_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name, or a number mapped to a color by bar.marker.line.colorscale, or a vector of these. go_bar_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name. go_bar_selected_textfont(): Sets the text font color of selected points. Color: a CSS color string or R color name. go_bar_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name. go_bar_unselected_textfont(): Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.
copy_ystyle	Logical.
symmetric	Determines whether or not the error bars have the same length in both direction (top/bottom for vertical bars, left/right for horizontal bars. Logical.
thickness	go_bar_error_x(), go_bar_error_y(): Sets the thickness (in px) of the error bars. Number in the interval [0, Inf]. go_bar_marker_colorbar(): Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].
traceref	Integer in the interval [0, Inf].

`tracerefminus` Integer in the interval [0, Inf].

`type` Determines the rule used to generate the error bars. If "constant", the bar lengths are of a constant value. Set this constant in ‘value‘. If "percent", the bar lengths correspond to a percentage of underlying data. Set this percentage in ‘value‘. If "sqrt", the bar lengths correspond to the square of the underlying data. If "data", the bar lengths are set with data set ‘array‘. One of `"percent"`, `"constant"`, `"sqrt"`, `"data"`.

`value` `go_bar_error_x()`, `go_bar_error_y()`: Sets the value of either the percentage (if ‘type‘ is set to "percent") or the constant (if ‘type‘ is set to "constant") corresponding to the lengths of the error bars. Number in the interval [0, Inf].

`go_bar_marker_colorbar_tickformatstop()`: string - dtickformat for described zoom level, the same as "tickformat" String.

`valueminus` Sets the value of either the percentage (if ‘type‘ is set to "percent") or the constant (if ‘type‘ is set to "constant") corresponding to the lengths of the error bars in the bottom (left) direction for vertical (horizontal) bars Number in the interval [0, Inf].

`align` Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

`bgcolor` `go_bar_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`go_bar_marker_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

`go_bar_marker_pattern()`: When there is no colorscale sets the color of background pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "overlay". Otherwise, defaults to a transparent background. Color: a CSS color string or R color name, or a vector of these.

`bordercolor` `go_bar_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`go_bar_marker_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

`font` `go_bar_hoverlabel()`: Sets the font used in hover labels. Named list or `go_bar_hoverlabel_font()` object.

`go_bar_legendgrouptitle()`: Sets this legend group’s title font. Named list or `go_bar_legendgrouptitle_font()` object.

`go_bar_marker_colorbar_title()`: Sets this color bar’s title font. Named list or `go_bar_marker_colorbar_title_font()` object.

`namelength` Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to ‘namelength - 3‘ characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

`showarrow` Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

family — go_bar_hoverlabel_font(), go_bar_insidetextfont(), go_bar_outsidetextfont(), go_bar_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

go_bar_legendgrouptitle_font(), go_bar_marker_colorbar_tickfont(), go_bar_marker_colorbar_title_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition — go_bar_hoverlabel_font(), go_bar_insidetextfont(), go_bar_outsidetextfont(), go_bar_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_bar_legendgrouptitle_font(), go_bar_marker_colorbar_tickfont(), go_bar_marker_colorbar_title_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow — go_bar_hoverlabel_font(), go_bar_insidetextfont(), go_bar_outsidetextfont(), go_bar_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_bar_legendgrouptitle_font(), go_bar_marker_colorbar_tickfont(), go_bar_marker_colorbar_title_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size — go_bar_hoverlabel_font(), go_bar_insidetextfont(), go_bar_outsidetextfont(), go_bar_textfont(): Number in the interval [1, Inf], or a vector of these.

go_bar_legendgrouptitle_font(), go_bar_marker_colorbar_tickfont(), go_bar_marker_colorbar_title_font(): Number in the interval [1, Inf].

go_bar_marker_pattern(): Sets the size of unit squares of the pattern fill in pixels, which corresponds to the interval of repetition of the pattern. Number in the interval [0, Inf], or a vector of these.

style — go_bar_hoverlabel_font(), go_bar_insidetextfont(), go_bar_outsidetextfont(), go_bar_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_bar_legendgrouptitle_font(), go_bar_marker_colorbar_tickfont(), go_bar_marker_colorbar_title_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase — go_bar_hoverlabel_font(), go_bar_insidetextfont(), go_bar_outsidetextfont(), go_bar_textfont(): Sets capitalization of text. It can be used to make text ap-

pear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_bar_legendgrouptitle_font(), go_bar_marker_colorbar_tickfont(), go_bar_marker_colorbar_title_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant

go_bar_hoverlabel_font(), go_bar_insidetextfont(), go_bar_outsidetextfont(), go_bar_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_bar_legendgrouptitle_font(), go_bar_marker_colorbar_tickfont(), go_bar_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight

go_bar_hoverlabel_font(), go_bar_insidetextfont(), go_bar_outsidetextfont(), go_bar_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_bar_legendgrouptitle_font(), go_bar_marker_colorbar_tickfont(), go_bar_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

autocolorscale

go_bar_marker(): Determines whether the colorscale is a default palette ('autocolorscale: true') or the palette determined by 'marker.colorscale'. Has an effect only if in 'marker.color' is set to a numerical array. In case 'colorscale' is unspecified or 'autocolorscale' is true, the default palette will be chosen according to whether numbers in the 'color' array are all positive, all negative or mixed. Logical.

go_bar_marker_line(): Determines whether the colorscale is a default palette ('autocolorscale: true') or the palette determined by 'marker.line.colorscale'. Has an effect only if in 'marker.line.color' is set to a numerical array. In case 'colorscale' is unspecified or 'autocolorscale' is true, the default palette will be chosen according to whether numbers in the 'color' array are all positive, all negative or mixed. Logical.

cauto

go_bar_marker(): Determines whether or not the color domain is computed with respect to the input data (here in 'marker.color') or the bounds set in 'marker.cmin' and 'marker.cmax' Has an effect only if in 'marker.color' is set to a numerical array. Defaults to 'false' when 'marker.cmin' and 'marker.cmax' are set by the user. Logical.

go_bar_marker_line(): Determines whether or not the color domain is computed with respect to the input data (here in 'marker.line.color') or the bounds set in 'marker.line.cmin' and 'marker.line.cmax' Has an effect only if in 'marker.line.color' is set to a numerical array. Defaults to 'false' when 'marker.line.cmin' and 'marker.line.cmax' are set by the user. Logical.

cmax

go_bar_marker(): Sets the upper bound of the color domain. Has an effect only if in 'marker.color' is set to a numerical array. Value should have the same units as in 'marker.color' and if set, 'marker.cmin' must be set as well. Number.

go_bar_marker_line(): Sets the upper bound of the color domain. Has an effect only if in 'marker.line.color' is set to a numerical array. Value should

have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

`cmid` `go_bar_marker()`: Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

`go_bar_marker_line()`: Sets the mid-point of the color domain by scaling ‘marker.line.cmin‘ and/or ‘marker.line.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘. Has no effect when ‘marker.line.cauto‘ is ‘false‘. Number.

`cmin` `go_bar_marker()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmax‘ must be set as well. Number.

`go_bar_marker_line()`: Sets the lower bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmax‘ must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_bar_marker_colorbar()` object.

`colorscale` `go_bar_marker()`: Sets the colorscale. Has an effect only if in ‘marker.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.cmin‘ and ‘marker.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Re
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`go_bar_marker_line()`: Sets the colorscale. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘marker.line.cmin‘ and ‘marker.line.cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

cornerradius
: Sets the rounding of corners. May be an integer number of pixels, or a percentage of bar width (as a string ending in %). Defaults to 'layout.barcornerradius'. In stack or relative barmode, the first trace to set cornerradius is used for the whole stack. Any value.

line
: Named list or `go_bar_marker_line()` object.

pattern
: Sets the pattern within the marker. Named list or `go_bar_marker_pattern()` object.

reversescale
: `go_bar_marker()`: Reverses the color mapping if true. Has an effect only if in 'marker.color' is set to a numerical array. If true, 'marker.cmin' will correspond to the last color in the array and 'marker.cmax' will correspond to the first color. Logical.

  `go_bar_marker_line()`: Reverses the color mapping if true. Has an effect only if in 'marker.line.color' is set to a numerical array. If true, 'marker.line.cmin' will correspond to the last color in the array and 'marker.line.cmax' will correspond to the first color. Logical.

showscale
: Determines whether or not a colorbar is displayed for this trace. Has an effect only if in 'marker.color' is set to a numerical array. Logical.

borderwidth
: Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

dtick
: Sets the step in-between ticks on this color bar. Use with 'tick0'. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

exponentformat
: Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

labelalias
: Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

len
: Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

lenmode
: Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

minexponent
: Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

nticks Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

outlinecolor Sets the axis line color. Color: a CSS color string or R color name.

outlinewidth Sets the width (in px) of the axis line. Number in the interval [0, Inf].

separatethousands
If "true", even 4-digit integers are separated Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticksuffix Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

thicknessmode Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

tick0 Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar’s tick label font Named list or `go_bar_marker_colorbar_tickfont()` object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops
List of named lists or `go_bar_marker_colorbar_tickformatstop()` objects.

ticklabeloverflow
Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition`
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep`
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen`
: Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode`
: Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix`
: Sets a tick label prefix. String.

`ticks`
: Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix`
: Sets a tick label suffix. String.

`ticktext`
: Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals`
: Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth`
: Sets the tick width (in px). Number in the interval [0, Inf].

`title`
: String (the title text), named list or `go_bar_marker_colorbar_title()` object.

`xanchor`
: Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad`
: Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref`
: Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`yanchor`
: Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

side — Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.

fgcolor — When there is no colorscale sets the color of foreground pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "replace". Otherwise, defaults to dark grey or white to increase contrast with the ‘bgcolor‘. Color: a CSS color string or R color name, or a vector of these.

fgopacity — Sets the opacity of the foreground pattern fill. Defaults to a 0.5 when ‘fillmode‘ is "overlay". Otherwise, defaults to 1. Number in the interval [0, 1].

fillmode — Determines whether ‘marker.color‘ should be used as a default to ‘bgcolor‘ or a ‘fgcolor‘. One of `"replace"`, `"overlay"`.

path — Sets a custom path for pattern fill. Use with no ‘shape‘ or ‘solidity‘, provide an SVG path string for the regions of the square from (0,0) to (‘size‘,‘size‘) to color. String, or a vector of these.

shape — Sets the shape of the pattern fill. By default, no pattern is used for filling the area. One of `""`, `"/"`, `"\\"`, `"x"`, `"-"`, `"|"`, `"+"`, `"."`, or a vector of these.

solidity — Sets the solidity of the pattern fill. Solidity is roughly the fraction of the area filled by the pattern. Solidity of 0 shows only the background color without pattern and solidty of 1 shows only the foreground color without pattern. Number in the interval [0, 1], or a vector of these.

fig — A `go_figure` object.

data — A data frame in which formula properties (e.g. `x = ~col`, `marker_size = ~col`) are evaluated. Default: the data given to `go_figure()`. Only the evaluated values are added to the trace. A `crosstalk::SharedData` works the same way, and links the trace to its crosstalk group (see `highlight()`).

row — Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.
secondary_y	If TRUE, use the secondary y axis of the subplot.

Details

go_bar() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_bar() creates an object and adds it to a figure.

Nested objects documented here:

go_bar_error_x() bar.error_x

go_bar_error_y() bar.error_y

go_bar_hoverlabel() bar.hoverlabel

go_bar_hoverlabel_font() bar.hoverlabel.font: Sets the font used in hover labels.

go_bar_insidetextfont() bar.insidetextfont: Sets the font used for ‘text‘ lying inside the bar.

go_bar_legendgrouptitle() bar.legendgrouptitle

go_bar_legendgrouptitle_font() bar.legendgrouptitle.font: Sets this legend group’s title font.

go_bar_marker() bar.marker

go_bar_marker_colorbar() bar.marker.colorbar

go_bar_marker_colorbar_tickfont() bar.marker.colorbar.tickfont: Sets the color bar’s tick label font

go_bar_marker_colorbar_tickformatstop() bar.marker.colorbar.tickformatstop

go_bar_marker_colorbar_title() bar.marker.colorbar.title

go_bar_marker_colorbar_title_font() bar.marker.colorbar.title.font: Sets this color bar’s title font.

go_bar_marker_line() bar.marker.line

go_bar_marker_pattern() bar.marker.pattern: Sets the pattern within the marker.

go_bar_outsidetextfont() bar.outsidetextfont: Sets the font used for ‘text‘ lying outside the bar.

go_bar_selected() bar.selected

go_bar_selected_marker() bar.selected.marker

go_bar_selected_textfont() bar.selected.textfont

go_bar_textfont() bar.textfont: Sets the font used for ‘text‘.

go_bar_unselected() bar.unselected

go_bar_unselected_marker() bar.unselected.marker

go_bar_unselected_textfont() bar.unselected.textfont

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_bar(x = 1:3, y = c(2, 1, 3), marker_color = "red")
```

`go_barpolar` *Barpolar traces*

Description

The data visualized by the radial span of the bars is set in ‘r‘

Usage

```
go_barpolar(
  arg = NULL,
  ...,
  base = NULL,
  customdata = NULL,
  dr = NULL,
  dtheta = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  offset = NULL,
  opacity = NULL,
  r = NULL,
  r0 = NULL,
  selected = NULL,
```

```
  selectedpoints = NULL,
  showlegend = NULL,
  subplot = NULL,
  text = NULL,
  theta = NULL,
  theta0 = NULL,
  thetaunit = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_barpolar_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_barpolar_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_barpolar_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
```

```
)

go_barpolar_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_barpolar_marker(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  line = NULL,
  opacity = NULL,
  pattern = NULL,
  reversescale = NULL,
  showscale = NULL,
  skip_invalid = FALSE
)

go_barpolar_marker_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
```

```
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_barpolar_marker_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
```

```
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_barpolar_marker_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_barpolar_marker_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_barpolar_marker_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_barpolar_marker_line(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
```

```
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  color = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  reversescale = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_barpolar_marker_pattern(
  arg = NULL,
  ...,
  bgcolor = NULL,
  fgcolor = NULL,
  fgopacity = NULL,
  fillmode = NULL,
  path = NULL,
  shape = NULL,
  size = NULL,
  solidity = NULL,
  skip_invalid = FALSE
)

go_barpolar_selected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_barpolar_selected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  skip_invalid = FALSE
)

go_barpolar_selected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)
```

```
go_barpolar_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  textfont = NULL,
  skip_invalid = FALSE
)

go_barpolar_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  skip_invalid = FALSE
)

go_barpolar_unselected_textfont(
  arg = NULL,
  ...,
  color = NULL,
  skip_invalid = FALSE
)

add_barpolar(fig, ...)

## Default S3 method:
add_barpolar(
  fig,
  ...,
  data = NULL,
  base = NULL,
  customdata = NULL,
  dr = NULL,
  dtheta = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
```

```
  offset = NULL,
  opacity = NULL,
  r = NULL,
  r0 = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  subplot = NULL,
  text = NULL,
  theta = NULL,
  theta0 = NULL,
  thetaunit = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  width = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
base	Sets where the bar base is drawn (in radial axis units). In "stack" barmode, traces that set "base" will be excluded and drawn in "overlay" mode instead. Any value.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
dr	Sets the r coordinate step. Number.
dtheta	Sets the theta coordinate step. By default, the ‘dtheta‘ step equals the subplot's period divided by the length of the ‘r‘ coordinates. Number.
hoverinfo	Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of `"r"`, `"theta"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.
hoverlabel	Named list or `go_barpolar_hoverlabel()` object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_},

{%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Same as 'text'. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_barpolar_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`marker`
: `go_barpolar()`: Named list or `go_barpolar_marker()` object.

`go_barpolar_selected()`: Named list or `go_barpolar_selected_marker()` object.

	go_barpolar_unselected(): Named list or go_barpolar_unselected_marker() object.
meta	Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.
name	go_barpolar(): Sets the trace name. The trace name appears as the legend item and on hover. String. go_barpolar_marker_colorbar_tickformatstop(): When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.
offset	Shifts the angular position where the bar is drawn (in "thetatunit" units). Number, or a vector of these.
opacity	go_barpolar(): Sets the opacity of the trace. Number in the interval [0, 1]. go_barpolar_marker(): Sets the opacity of the bars. Number in the interval [0, 1], or a vector of these. go_barpolar_selected_marker(): Sets the marker opacity of selected points. Number in the interval [0, 1]. go_barpolar_unselected_marker(): Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].
r	Sets the radial coordinates Data array: a vector, factor, Date/POSIXct vector, matrix or list.
r0	Alternate to ‘r‘. Builds a linear space of r coordinates. Use with ‘dr‘ where ‘r0‘ is the starting coordinate and ‘dr‘ the step. Any value.
selected	Named list or go_barpolar_selected() object.
selectedpoints	Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.
showlegend	Determines whether or not an item corresponding to this trace is shown in the legend. Logical.
subplot	Sets a reference between this trace’s data coordinates and a polar subplot. If "polar" (the default value), the data refer to ‘layout.polar‘. If "polar2", the data refer to ‘layout.polar2‘, and so on. Subplot id: "polar", "polar2", "polar3", ...

text	go_barpolar(): Sets hover text elements associated with each bar. If a single string, the same string appears over all bars. If an array of string, the items are mapped in order to the this trace's coordinates. String, or a vector of these. go_barpolar_legendgrouptitle(): Sets the title of the legend group. String. go_barpolar_marker_colorbar_title(): Sets the title of the color bar. String.
theta	Sets the angular coordinates Data array: a vector, factor, Date/POSIXct vector, matrix or list.
theta0	Alternate to 'theta'. Builds a linear space of theta coordinates. Use with 'dtheta' where 'theta0' is the starting coordinate and 'dtheta' the step. Any value.
thetaunit	Sets the unit of input "theta" values. Has an effect only when on "linear" angular axes. One of "radians", "degrees", "gradians".
uid	Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.
uirevision	Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.
unselected	Named list or go_barpolar_unselected() object.
visible	Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.
width	go_barpolar(): Sets the bar angular width (in "thetaunit" units). Number in the interval [0, Inf], or a vector of these. go_barpolar_marker_line(): Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
align	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.
bgcolor	go_barpolar_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these. go_barpolar_marker_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name. go_barpolar_marker_pattern(): When there is no colorscale sets the color of background pattern fill. Defaults to a 'marker.color' background when 'fillmode' is "overlay". Otherwise, defaults to a transparent background. Color: a CSS color string or R color name, or a vector of these.

bordercolor
: go_barpolar_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.
go_barpolar_marker_colorbar(): Sets the axis line color. Color: a CSS color string or R color name.

font
: go_barpolar_hoverlabel(): Sets the font used in hover labels. Named list or go_barpolar_hoverlabel_font() object.
go_barpolar_legendgrouptitle(): Sets this legend group's title font. Named list or go_barpolar_legendgrouptitle_font() object.
go_barpolar_marker_colorbar_title(): Sets this color bar's title font. Named list or go_barpolar_marker_colorbar_title_font() object.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color
: go_barpolar_hoverlabel_font(): Color: a CSS color string or R color name, or a vector of these.
go_barpolar_legendgrouptitle_font(), go_barpolar_marker_colorbar_tickfont(), go_barpolar_marker_colorbar_title_font(): Color: a CSS color string or R color name.
go_barpolar_marker(): Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.cmin' and 'marker.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by barpolar.marker.colorscale, or a vector of these.
go_barpolar_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to 'marker.line.cmin' and 'marker.line.cmax' if set. Color: a CSS color string or R color name, or a number mapped to a color by barpolar.marker.line.colorscale, or a vector of these.
go_barpolar_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name.
go_barpolar_selected_textfont(): Sets the text font color of selected points. Color: a CSS color string or R color name.
go_barpolar_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.
go_barpolar_unselected_textfont(): Sets the text font color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

family
: go_barpolar_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated

by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

`go_barpolar_legendgrouptitle_font()`, `go_barpolar_marker_colorbar_tickfont()`, `go_barpolar_marker_colorbar_title_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

`lineposition`

`go_barpolar_hoverlabel_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

`go_barpolar_legendgrouptitle_font()`, `go_barpolar_marker_colorbar_tickfont()`, `go_barpolar_marker_colorbar_title_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

`shadow`

`go_barpolar_hoverlabel_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

`go_barpolar_legendgrouptitle_font()`, `go_barpolar_marker_colorbar_tickfont()`, `go_barpolar_marker_colorbar_title_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

`size`

`go_barpolar_hoverlabel_font()`: Number in the interval [1, Inf], or a vector of these.

`go_barpolar_legendgrouptitle_font()`, `go_barpolar_marker_colorbar_tickfont()`, `go_barpolar_marker_colorbar_title_font()`: Number in the interval [1, Inf].

`go_barpolar_marker_pattern()`: Sets the size of unit squares of the pattern fill in pixels, which corresponds to the interval of repetition of the pattern. Number in the interval [0, Inf], or a vector of these.

`style`

`go_barpolar_hoverlabel_font()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`, or a vector of these.

`go_barpolar_legendgrouptitle_font()`, `go_barpolar_marker_colorbar_tickfont()`, `go_barpolar_marker_colorbar_title_font()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

`textcase`

`go_barpolar_hoverlabel_font()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`, or a vector of these.

`go_barpolar_legendgrouptitle_font()`, `go_barpolar_marker_colorbar_tickfont()`, `go_barpolar_marker_colorbar_title_font()`: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.

variant
: go_barpolar_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

  go_barpolar_legendgrouptitle_font(), go_barpolar_marker_colorbar_tickfont(), go_barpolar_marker_colorbar_title_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_barpolar_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

  go_barpolar_legendgrouptitle_font(), go_barpolar_marker_colorbar_tickfont(), go_barpolar_marker_colorbar_title_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

autocolorscale
: go_barpolar_marker(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.colorscale‘. Has an effect only if in ‘marker.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

  go_barpolar_marker_line(): Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘marker.line.colorscale‘. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.

cauto
: go_barpolar_marker(): Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.color‘) or the bounds set in ‘marker.cmin‘ and ‘marker.cmax‘ Has an effect only if in ‘marker.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.cmin‘ and ‘marker.cmax‘ are set by the user. Logical.

  go_barpolar_marker_line(): Determines whether or not the color domain is computed with respect to the input data (here in ‘marker.line.color‘) or the bounds set in ‘marker.line.cmin‘ and ‘marker.line.cmax‘ Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Defaults to ‘false‘ when ‘marker.line.cmin‘ and ‘marker.line.cmax‘ are set by the user. Logical.

cmax
: go_barpolar_marker(): Sets the upper bound of the color domain. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘ and if set, ‘marker.cmin‘ must be set as well. Number.

  go_barpolar_marker_line(): Sets the upper bound of the color domain. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. Value should have the same units as in ‘marker.line.color‘ and if set, ‘marker.line.cmin‘ must be set as well. Number.

cmid
: go_barpolar_marker(): Sets the mid-point of the color domain by scaling ‘marker.cmin‘ and/or ‘marker.cmax‘ to be equidistant to this point. Has an effect only if in ‘marker.color‘ is set to a numerical array. Value should have the same units as in ‘marker.color‘. Has no effect when ‘marker.cauto‘ is ‘false‘. Number.

go_barpolar_marker_line(): Sets the mid-point of the color domain by scaling 'marker.line.cmin' and/or 'marker.line.cmax' to be equidistant to this point. Has an effect only if in 'marker.line.color' is set to a numerical array. Value should have the same units as in 'marker.line.color'. Has no effect when 'marker.line.cauto' is 'false'. Number.

`cmin` go_barpolar_marker(): Sets the lower bound of the color domain. Has an effect only if in 'marker.color' is set to a numerical array. Value should have the same units as in 'marker.color' and if set, 'marker.cmax' must be set as well. Number.

go_barpolar_marker_line(): Sets the lower bound of the color domain. Has an effect only if in 'marker.line.color' is set to a numerical array. Value should have the same units as in 'marker.line.color' and if set, 'marker.line.cmax' must be set as well. Number.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under 'layout.coloraxis', 'layout.coloraxis2', etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_barpolar_marker_colorbar()` object.

`colorscale` go_barpolar_marker(): Sets the colorscale. Has an effect only if in 'marker.color' is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the bounds of the colorscale in color space, use 'marker.cmin' and 'marker.cmax'. Alternatively, 'colorscale' may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Re
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

go_barpolar_marker_line(): Sets the colorscale. Has an effect only if in 'marker.line.color' is set to a numerical array. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, '[[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]'. To control the bounds of the colorscale in color space, use 'marker.line.cmin' and 'marker.line.cmax'. Alternatively, 'colorscale' may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,H
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`line` Named list or `go_barpolar_marker_line()` object.

`pattern` Sets the pattern within the marker. Named list or `go_barpolar_marker_pattern()` object.

`reversescale` go_barpolar_marker(): Reverses the color mapping if true. Has an effect only if in 'marker.color' is set to a numerical array. If true, 'marker.cmin' will

correspond to the last color in the array and ‘marker.cmax‘ will correspond to the first color. Logical.

go_barpolar_marker_line(): Reverses the color mapping if true. Has an effect only if in ‘marker.line.color‘ is set to a numerical array. If true, ‘marker.line.cmin‘ will correspond to the last color in the array and ‘marker.line.cmax‘ will correspond to the first color. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Has an effect only if in ‘marker.color‘ is set to a numerical array. Logical.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands`
If "true", even 4-digit integers are separated Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticksuffix Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

thickness Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

tick0 Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar’s tick label font Named list or `go_barpolar_marker_colorbar_tickfont()` object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or `go_barpolar_marker_colorbar_tickformatstop()` objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

ticklabelposition Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is

"v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelstep — Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

ticklen — Sets the tick length (in px). Number in the interval [0, Inf].

tickmode — Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of "auto", "linear", "array".

tickprefix — Sets a tick label prefix. String.

ticks — Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of "outside", "inside", "".

ticksuffix — Sets a tick label suffix. String.

ticktext — Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals — Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth — Sets the tick width (in px). Number in the interval [0, Inf].

title — String (the title text), named list or go_barpolar_marker_colorbar_title() object.

x — Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

xanchor — Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of "left", "center", "right".

xpad — Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".

y Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

yanchor Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

ypad Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

dtickrange range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname
Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value string - dtickformat for described zoom level, the same as "tickformat" String.

side Determines the location of color bar’s title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.

fgcolor When there is no colorscale sets the color of foreground pattern fill. Defaults to a ‘marker.color‘ background when ‘fillmode‘ is "replace". Otherwise, defaults to dark grey or white to increase contrast with the ‘bgcolor‘. Color: a CSS color string or R color name, or a vector of these.

fgopacity Sets the opacity of the foreground pattern fill. Defaults to a 0.5 when ‘fillmode‘ is "overlay". Otherwise, defaults to 1. Number in the interval [0, 1].

fillmode Determines whether ‘marker.color‘ should be used as a default to ‘bgcolor‘ or a ‘fgcolor‘. One of `"replace"`, `"overlay"`.

path Sets a custom path for pattern fill. Use with no ‘shape‘ or ‘solidity‘, provide an SVG path string for the regions of the square from (0,0) to (‘size‘,‘size‘) to color. String, or a vector of these.

shape Sets the shape of the pattern fill. By default, no pattern is used for filling the area. One of `""`, `"/"`, `"\\"`, `"x"`, `"-"`, `"|"`, `"+"`, `"."`, or a vector of these.

solidity	Sets the solidity of the pattern fill. Solidity is roughly the fraction of the area filled by the pattern. Solidity of 0 shows only the background color without pattern and solidty of 1 shows only the foreground color without pattern. Number in the interval [0, 1], or a vector of these.
textfont	go_barpolar_selected(): Named list or go_barpolar_selected_textfont() object. go_barpolar_unselected(): Named list or go_barpolar_unselected_textfont() object.
fig	A go_figure object.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
row	Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

Details

go_barpolar() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_barpolar() creates an object and adds it to a figure.

Nested objects documented here:

go_barpolar_hoverlabel() barpolar.hoverlabel

go_barpolar_hoverlabel_font() barpolar.hoverlabel.font: Sets the font used in hover labels.

go_barpolar_legendgrouptitle() barpolar.legendgrouptitle

go_barpolar_legendgrouptitle_font() barpolar.legendgrouptitle.font: Sets this legend group's title font.

go_barpolar_marker() barpolar.marker

go_barpolar_marker_colorbar() barpolar.marker.colorbar

go_barpolar_marker_colorbar_tickfont() barpolar.marker.colorbar.tickfont: Sets the color bar's tick label font

go_barpolar_marker_colorbar_tickformatstop() barpolar.marker.colorbar.tickformatstop

go_barpolar_marker_colorbar_title() barpolar.marker.colorbar.title

go_barpolar_marker_colorbar_title_font() barpolar.marker.colorbar.title.font: Sets this color bar's title font.

go_barpolar_marker_line() barpolar.marker.line

go_barpolar_marker_pattern() barpolar.marker.pattern: Sets the pattern within the marker.

go_barpolar_selected() barpolar.selected

go_barpolar_selected_marker() barpolar.selected.marker

`go_barpolar_selected_textfont()` barpolar.selected.textfont

`go_barpolar_unselected()` barpolar.unselected

`go_barpolar_unselected_marker()` barpolar.unselected.marker

`go_barpolar_unselected_textfont()` barpolar.unselected.textfont

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_barpolar(marker_color = "red")
```

---

go_box *Box traces*

Description

Each box spans from quartile 1 (Q1) to quartile 3 (Q3). The second quartile (Q2, i.e. the median) is marked by a line inside the box. The fences grow outward from the boxes' edges, by default they span +/- 1.5 times the interquartile range (IQR: Q3-Q1), The sample mean and standard deviation as well as notches and the sample, outlier and suspected outliers points can be optionally added to the box plot. The values and positions corresponding to each boxes can be input using two signatures. The first signature expects users to supply the sample values in the ‘y‘ data array for vertical boxes (‘x‘ for horizontal boxes). By supplying an ‘x‘ (‘y‘) array, one box per distinct ‘x‘ (‘y‘) value is drawn If no ‘x‘ (‘y‘) vector is provided, a single box is drawn. In this case, the box is positioned with the trace ‘name‘ or with ‘x0‘ (‘y0‘) if provided. The second signature expects users to supply the boxes corresponding Q1, median and Q3 statistics in the ‘q1‘, ‘median‘ and ‘q3‘ data arrays respectively. Other box features relying on statistics namely ‘lowerfence‘, ‘upperfence‘, ‘notchspan‘ can be set directly by the users. To have plotly compute them or to show sample points besides the boxes, users can set the ‘y‘ data array for vertical boxes (‘x‘ for horizontal boxes) to a 2D array with the outer length corresponding to the number of boxes in the traces and the inner length corresponding the sample size.

Usage

```
go_box(
  arg = NULL,
  ...,
  alignmentgroup = NULL,
  boxmean = NULL,
  boxpoints = NULL,
```

```
    customdata = NULL,
    dx = NULL,
    dy = NULL,
    fillcolor = NULL,
    hoverinfo = NULL,
    hoverlabel = NULL,
    hoveron = NULL,
    hovertemplate = NULL,
    hovertemplatefallback = NULL,
    hovertext = NULL,
    ids = NULL,
    jitter = NULL,
    legend = NULL,
    legendgroup = NULL,
    legendgrouptitle = NULL,
    legendrank = NULL,
    legendwidth = NULL,
    line = NULL,
    lowerfence = NULL,
    marker = NULL,
    mean = NULL,
    median = NULL,
    meta = NULL,
    name = NULL,
    notched = NULL,
    notchspan = NULL,
    notchwidth = NULL,
    offsetgroup = NULL,
    opacity = NULL,
    orientation = NULL,
    pointpos = NULL,
    q1 = NULL,
    q3 = NULL,
    quartilemethod = NULL,
    sd = NULL,
    sdmultiple = NULL,
    selected = NULL,
    selectedpoints = NULL,
    showlegend = NULL,
    showwhiskers = NULL,
    sizemode = NULL,
    text = NULL,
    uid = NULL,
    uirevision = NULL,
    unselected = NULL,
    upperfence = NULL,
    visible = NULL,
    whiskerwidth = NULL,
```

```
  width = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  yperiod = NULL,
  yperiod0 = NULL,
  yperiodalignment = NULL,
  zorder = NULL,
  skip_invalid = FALSE
)

go_box_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_box_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

```
go_box_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_box_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_box_line(arg = NULL, ..., color = NULL, width = NULL, skip_invalid = FALSE)

go_box_marker(
  arg = NULL,
  ...,
  angle = NULL,
  color = NULL,
  line = NULL,
  opacity = NULL,
  outliercolor = NULL,
  size = NULL,
  symbol = NULL,
  skip_invalid = FALSE
)

go_box_marker_line(
  arg = NULL,
  ...,
  color = NULL,
  outliercolor = NULL,
  outlierwidth = NULL,
  width = NULL,
  skip_invalid = FALSE
)
```

```
go_box_selected(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

go_box_selected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

go_box_unselected(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

go_box_unselected_marker(
  arg = NULL,
  ...,
  color = NULL,
  opacity = NULL,
  size = NULL,
  skip_invalid = FALSE
)

add_box(fig, ...)

## Default S3 method:
add_box(
  fig,
  ...,
  data = NULL,
  alignmentgroup = NULL,
  boxmean = NULL,
  boxpoints = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoveron = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  jitter = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
```

```
    legendwidth = NULL,
    line = NULL,
    lowerfence = NULL,
    marker = NULL,
    mean = NULL,
    median = NULL,
    meta = NULL,
    name = NULL,
    notched = NULL,
    notchspan = NULL,
    notchwidth = NULL,
    offsetgroup = NULL,
    opacity = NULL,
    orientation = NULL,
    pointpos = NULL,
    q1 = NULL,
    q3 = NULL,
    quartilemethod = NULL,
    sd = NULL,
    sdmultiple = NULL,
    selected = NULL,
    selectedpoints = NULL,
    showlegend = NULL,
    showwhiskers = NULL,
    sizemode = NULL,
    text = NULL,
    uid = NULL,
    uirevision = NULL,
    unselected = NULL,
    upperfence = NULL,
    visible = NULL,
    whiskerwidth = NULL,
    width = NULL,
    x = NULL,
    x0 = NULL,
    xaxis = NULL,
    xcalendar = NULL,
    xhoverformat = NULL,
    xperiod = NULL,
    xperiod0 = NULL,
    xperiodalignment = NULL,
    y = NULL,
    y0 = NULL,
    yaxis = NULL,
    ycalendar = NULL,
    yhoverformat = NULL,
    yperiod = NULL,
    yperiod0 = NULL,
```

```
  yperiodalignment = NULL,
  zorder = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
alignmentgroup	Set several traces linked to the same position axis or matching axes to the same alignmentgroup. This controls whether bars compute their positional range dependently or independently. String.
boxmean	If TRUE, the mean of the box(es)' underlying distribution is drawn as a dashed line inside the box(es). If "sd" the standard deviation is also drawn. Defaults to TRUE when 'mean' is set. Defaults to "sd" when 'sd' is set Otherwise defaults to FALSE. One of `"sd"`, `TRUE`, `FALSE`.
boxpoints	If "outliers", only the sample points lying outside the whiskers are shown If "suspectedoutliers", the outlier points are shown and points either less than 4"Q1-3"Q3 or greater than 4"Q3-3"Q1 are highlighted (see 'outliercolor') If "all", all sample points are shown If FALSE, only the box(es) are shown with no sample points Defaults to "suspectedoutliers" when 'marker.outliercolor' or 'marker.line.outliercolor' is set. Defaults to "all" under the q1/median/q3 signature. Otherwise defaults to "outliers". One of `"all"`, `"outliers"`, `"suspectedoutliers"`, `FALSE`.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
dx	Sets the x coordinate step for multi-box traces set using q1/median/q3. Number.
dy	Sets the y coordinate step for multi-box traces set using q1/median/q3. Number.
fillcolor	Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name.
hoverinfo	Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.
hoverlabel	Named list or `go_box_hoverlabel()` object.
hoveron	Do the hover effects highlight individual boxes or sample points or both? Any combination of `"boxes"`, `"points"` joined with `"+"`.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback`
Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Same as 'text'. String, or a vector of these.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`jitter` Sets the amount of jitter in the sample points drawn. If 0, the sample points align along the distribution axis. If 1, the sample points are drawn in a random jitter of width equal to the width of the box(es). Number in the interval [0, 1].

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
Named list or `go_box_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line` `go_box()`: Named list or `go_box_line()` object.

`go_box_marker()`: Named list or `go_box_marker_line()` object.

`lowerfence` Sets the lower fence values. There should be as many items as the number of boxes desired. This attribute has effect only under the q1/median/q3 signature. If 'lowerfence' is not provided but a sample (in 'y' or 'x') is set, we compute the lower as the last sample point below 1.5 times the IQR. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`marker` `go_box()`: Named list or `go_box_marker()` object.

`go_box_selected()`: Named list or `go_box_selected_marker()` object.

`go_box_unselected()`: Named list or `go_box_unselected_marker()` object.

`mean` Sets the mean values. There should be as many items as the number of boxes desired. This attribute has effect only under the q1/median/q3 signature. If 'mean' is not provided but a sample (in 'y' or 'x') is set, we compute the mean for each box using the sample values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`median` Sets the median values. There should be as many items as the number of boxes desired. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

`name` Sets the trace name. The trace name appears as the legend item and on hover. For box traces, the name will also be used for the position coordinate, if 'x' and 'x0' ('y' and 'y0' if horizontal) are missing and the position axis is categorical String.

`notched` Determines whether or not notches are drawn. Notches displays a confidence interval around the median. We compute the confidence interval as median +/- 1.57 * IQR / sqrt(N), where IQR is the interquartile range and N is the sample size. If two boxes' notches do not overlap there is 95% confidence their medians differ. See https://sites.google.com/site/davidsstatistics/home/notched-box-plots for more info. Defaults to FALSE unless 'notchwidth' or 'notchspan' is set. Logical.

`notchspan` Sets the notch span from the boxes' 'median' values. There should be as many items as the number of boxes desired. This attribute has effect only under the q1/median/q3 signature. If 'notchspan' is not provided but a sample (in 'y' or 'x') is set, we compute it as 1.57 * IQR / sqrt(N), where N is the sample size. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`notchwidth` Sets the width of the notches relative to the box width. For example, with 0, the notches are as wide as the box(es). Number in the interval [0, 0.5].

`offsetgroup` Set several traces linked to the same position axis or matching axes to the same offsetgroup where bars of the same position coordinate will line up. String.

`opacity` `go_box()`: Sets the opacity of the trace. Number in the interval [0, 1].

`go_box_marker()`: Sets the marker opacity. Number in the interval [0, 1].

`go_box_selected_marker()`: Sets the marker opacity of selected points. Number in the interval [0, 1].

`go_box_unselected_marker()`: Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

`orientation` Sets the orientation of the box(es). If "v" ("h"), the distribution is visualized along the vertical (horizontal). One of `"v"`, `"h"`.

`pointpos` Sets the position of the sample points in relation to the box(es). If 0, the sample points are places over the center of the box(es). Positive (negative) values correspond to positions to the right (left) for vertical boxes and above (below) for horizontal boxes Number in the interval [-2, 2].

`q1` Sets the Quartile 1 values. There should be as many items as the number of boxes desired. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`q3` Sets the Quartile 3 values. There should be as many items as the number of boxes desired. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`quartilemethod` Sets the method used to compute the sample's Q1 and Q3 quartiles. The "linear" method uses the 25th percentile for Q1 and 75th percentile for Q3 as computed using method #10 (listed on http://jse.amstat.org/v14n3/langford.html). The "exclusive" method uses the median to divide the ordered dataset into two halves if the sample is odd, it does not include the median in either half - Q1 is then the median of the lower half and Q3 the median of the upper half. The "inclusive" method also uses the median to divide the ordered dataset into two halves but if the sample is odd, it includes the median in both halves - Q1 is then the median of the lower half and Q3 the median of the upper half. One of `"linear"`, `"exclusive"`, `"inclusive"`.

`sd` Sets the standard deviation values. There should be as many items as the number of boxes desired. This attribute has effect only under the q1/median/q3 signature. If 'sd' is not provided but a sample (in 'y' or 'x') is set, we compute the standard deviation for each box using the sample values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`sdmultiple` Scales the box size when sizemode=sd Allowing boxes to be drawn across any stddev range For example 1-stddev, 3-stddev, 5-stddev Number in the interval [0, Inf].

`selected` Named list or `go_box_selected()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the 'unselected' are turned on for all points, whereas, any other non-array values means no selection all where the 'selected' and 'unselected' styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

showwhiskers Determines whether or not whiskers are visible. Defaults to true for ‘sizemode‘ "quartiles", false for "sd". Logical.

sizemode Sets the upper and lower bound for the boxes quartiles means box is drawn between Q1 and Q3 SD means the box is drawn between Mean +- Standard Deviation Argument sdmultiple (default 1) to scale the box size So it could be drawn 1-stddev, 3-stddev etc One of `"quartiles"`, `"sd"`.

text `go_box()`: Sets the text elements associated with each sample value. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (x,y) coordinates. To be seen, trace ‘hoverinfo‘ must contain a "text" flag. String, or a vector of these.

`go_box_legendgrouptitle()`: Sets the title of the legend group. String.

uid Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

uirevision Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(x|y)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

unselected Named list or `go_box_unselected()` object.

upperfence Sets the upper fence values. There should be as many items as the number of boxes desired. This attribute has effect only under the q1/median/q3 signature. If ‘upperfence‘ is not provided but a sample (in ‘y‘ or ‘x‘) is set, we compute the upper as the last sample point above 1.5 times the IQR. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

visible Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

whiskerwidth Sets the width of the whiskers relative to the box width. For example, with 1, the whiskers are as wide as the box(es). Number in the interval [0, 1].

width `go_box()`: Sets the width of the box in data coordinate If 0 (default value) the width is automatically selected based on the positions of other box traces in the same subplot. Number in the interval [0, Inf].

`go_box_line()`: Sets the width (in px) of line bounding the box(es). Number in the interval [0, Inf].

`go_box_marker_line()`: Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf].

x Sets the x sample data or coordinates. See overview for more info. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

x0 Sets the x coordinate for single-box traces or the starting coordinate for multi-box traces set using q1/median/q3. See overview for more info. Any value.

xaxis Sets a reference between this trace's x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to 'layout.xaxis'. If "x2", the x coordinates refer to 'layout.xaxis2', and so on. Subplot id: "x", "x2", "x3", ...

xcalendar Sets the calendar system to use with 'x' date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

xhoverformat Sets the hover text formatting rule for 'x' using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using 'xaxis.hoverformat'. String.

xperiod Only relevant when the axis 'type' is "date". Sets the period positioning in milliseconds or "M<n>" on the x axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case 'n' must be a positive integer. Any value.

xperiod0 Only relevant when the axis 'type' is "date". Sets the base for period positioning in milliseconds or date string on the x0 axis. When 'x0period' is round number of weeks, the 'x0period0' by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

xperiodalignment
Only relevant when the axis 'type' is "date". Sets the alignment of data points on the x axis. One of "start", "middle", "end".

y Sets the y sample data or coordinates. See overview for more info. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

y0 Sets the y coordinate for single-box traces or the starting coordinate for multi-box traces set using q1/median/q3. See overview for more info. Any value.

yaxis Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to 'layout.yaxis'. If "y2", the y coordinates refer to 'layout.yaxis2', and so on. Subplot id: "y", "y2", "y3", ...

ycalendar Sets the calendar system to use with 'y' date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

yhoverformat Sets the hover text formatting rule for 'y' using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using 'yaxis.hoverformat'. String.

yperiod — Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the y axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

yperiod0 — Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the y0 axis. When ‘y0period‘ is round number of weeks, the ‘y0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

yperiodalignment — Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the y axis. One of `"start"`, `"middle"`, `"end"`.

zorder — Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher ‘zorder‘ appear in front of those with lower ‘zorder‘. Integer.

skip_invalid — If `TRUE`, silently drop invalid properties instead of raising an error.

align — Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

bgcolor — Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

bordercolor — Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

font — `go_box_hoverlabel()`: Sets the font used in hover labels. Named list or `go_box_hoverlabel_font()` object.

`go_box_legendgrouptitle()`: Sets this legend group’s title font. Named list or `go_box_legendgrouptitle_font()` object.

namelength — Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to ‘namelength - 3‘ characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow — Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

color — `go_box_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

`go_box_legendgrouptitle_font()`: Color: a CSS color string or R color name.

`go_box_line()`: Sets the color of line bounding the box(es). Color: a CSS color string or R color name.

`go_box_marker()`: Sets the marker color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.cmin‘ and ‘marker.cmax‘ if set. Color: a CSS color string or R color name.

go_box_marker_line(): Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name.

go_box_selected_marker(): Sets the marker color of selected points. Color: a CSS color string or R color name.

go_box_unselected_marker(): Sets the marker color of unselected points, applied only when a selection exists. Color: a CSS color string or R color name.

family

go_box_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

go_box_legendgrouptitle_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition

go_box_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

go_box_legendgrouptitle_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow

go_box_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

go_box_legendgrouptitle_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size

go_box_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

go_box_legendgrouptitle_font(): Number in the interval [1, Inf].

go_box_marker(): Sets the marker size (in px). Number in the interval [0, Inf].

go_box_selected_marker(): Sets the marker size of selected points. Number in the interval [0, Inf].

go_box_unselected_marker(): Sets the marker size of unselected points, applied only when a selection exists. Number in the interval [0, Inf].

style

go_box_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

go_box_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
go_box_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

go_box_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
go_box_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

go_box_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
go_box_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

go_box_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

angle
Sets the marker angle in respect to 'angleref'. Angle in degrees.

outliercolor
go_box_marker(): Sets the color of the outlier sample points. Color: a CSS color string or R color name.

go_box_marker_line(): Sets the border line color of the outlier sample points. Defaults to marker.color Color: a CSS color string or R color name.

symbol
Sets the marker symbol type. Adding 100 is equivalent to appending "-open" to a symbol name. Adding 200 is equivalent to appending "-dot" to a symbol name. Adding 300 is equivalent to appending "-open-dot" or "dot-open" to a symbol name. One of "0", "circle", "100", "circle-open", "200", "circle-dot", "300", "circle-open-dot", "1", "square", "101", "square-open", "201", "square-dot", "301", "square-open-dot", "2", "diamond", "102", "diamond-open", "202", "diamond-dot", "302", "diamond-open-dot", "3", "cross", "103", "cross-open", "203", "cross-dot", "303", "cross-open-dot", "4", "x", "104", "x-open", "204", "x-dot", "304", "x-open-dot", "5", "triangle-up", "105", "triangle-up-open", "205", "triangle-up-dot", "305", "triangle-up-open-dot", "6", "triangle-down", "106", "triangle-down-open", "206", "triangle-down-dot", "306", "triangle-down-open-dot", "7", "triangle-left", "107", "triangle-left-open", "207", "triangle-left-dot", "307", "triangle-left-open-dot", "8", "triangle-right", "108", "triangle-right-open", "208", "triangle-right-dot", "308", "triangle-right-open-do "9", "triangle-ne", "109", "triangle-ne-open", "209", "triangle-ne-dot", "309", "triangle-ne-open-dot", "10", "triangle-se", "110", "triangle-se-open", "210", "triangle-se-dot", "310", "triangle-se-open-dot", "11", "triangle-sw", "111", "triangle-sw-open", "211", "triangle-sw-dot", "311", "triangle-sw-open-dot", "12", "triangle-nw", "112", "triangle-nw-open", "212", "triangle-nw-dot", "312", "triangle-nw-open-dot", "13", "pentagon", "113", "pentagon-open", "213", "pentagon-dot", "313", "pentagon-open-dot", "14", "hexagon", "114", "hexagon-open", "214", "hexagon-dot", "314", "hexagon-open-dot", "15", "hexagon2", "115", "hexagon2-open", "215", "hexagon2-dot", "315", "hexagon2-open-dot", "16", "octagon", "116", "octagon-open", "216",

"octagon-dot", "316", "octagon-open-dot", "17", "star", "117", "star-open",
"217", "star-dot", "317", "star-open-dot", "18", "hexagram", "118", "hexagram-open",
"218", "hexagram-dot", "318", "hexagram-open-dot", "19", "star-triangle-up",
"119", "star-triangle-up-open", "219", "star-triangle-up-dot", "319",
"star-triangle-up-open-dot", "20", "star-triangle-down", "120", "star-triangle-down-open
"220", "star-triangle-down-dot", "320", "star-triangle-down-open-dot",
"21", "star-square", "121", "star-square-open", "221", "star-square-dot",
"321", "star-square-open-dot", "22", "star-diamond", "122", "star-diamond-open",
"222", "star-diamond-dot", "322", "star-diamond-open-dot", "23", "diamond-tall",
"123", "diamond-tall-open", "223", "diamond-tall-dot", "323", "diamond-tall-open-dot",
"24", "diamond-wide", "124", "diamond-wide-open", "224", "diamond-wide-dot",
"324", "diamond-wide-open-dot", "25", "hourglass", "125", "hourglass-open",
"26", "bowtie", "126", "bowtie-open", "27", "circle-cross", "127", "circle-cross-open",
"28", "circle-x", "128", "circle-x-open", "29", "square-cross", "129",
"square-cross-open", "30", "square-x", "130", "square-x-open", "31",
"diamond-cross", "131", "diamond-cross-open", "32", "diamond-x", "132",
"diamond-x-open", "33", "cross-thin", "133", "cross-thin-open", "34",
"x-thin", "134", "x-thin-open", "35", "asterisk", "135", "asterisk-open",
"36", "hash", "136", "hash-open", "236", "hash-dot", "336", "hash-open-dot",
"37", "y-up", "137", "y-up-open", "38", "y-down", "138", "y-down-open",
"39", "y-left", "139", "y-left-open", "40", "y-right", "140", "y-right-open",
"41", "line-ew", "141", "line-ew-open", "42", "line-ns", "142", "line-ns-open",
"43", "line-ne", "143", "line-ne-open", "44", "line-nw", "144", "line-nw-open",
"45", "arrow-up", "145", "arrow-up-open", "46", "arrow-down", "146",
"arrow-down-open", "47", "arrow-left", "147", "arrow-left-open", "48",
"arrow-right", "148", "arrow-right-open", "49", "arrow-bar-up", "149",
"arrow-bar-up-open", "50", "arrow-bar-down", "150", "arrow-bar-down-open",
"51", "arrow-bar-left", "151", "arrow-bar-left-open", "52", "arrow-bar-right",
"152", "arrow-bar-right-open", "53", "arrow", "153", "arrow-open", "54",
"arrow-wide", "154", "arrow-wide-open", 0, 100, 200, 300, 1, 101, 201, 301,
2, 102, 202, 302, 3, 103, 203, 303, 4, 104, 204, 304, 5, 105, 205, 305, 6, 106,
206, 306, 7, 107, 207, 307, 8, 108, 208, 308, 9, 109, 209, 309, 10, 110, 210, 310,
11, 111, 211, 311, 12, 112, 212, 312, 13, 113, 213, 313, 14, 114, 214, 314, 15,
115, 215, 315, 16, 116, 216, 316, 17, 117, 217, 317, 18, 118, 218, 318, 19, 119,
219, 319, 20, 120, 220, 320, 21, 121, 221, 321, 22, 122, 222, 322, 23, 123, 223,
323, 24, 124, 224, 324, 25, 125, 26, 126, 27, 127, 28, 128, 29, 129, 30, 130, 31,
131, 32, 132, 33, 133, 34, 134, 35, 135, 36, 136, 236, 336, 37, 137, 38, 138, 39,
139, 40, 140, 41, 141, 42, 142, 43, 143, 44, 144, 45, 145, 46, 146, 47, 147, 48,
148, 49, 149, 50, 150, 51, 151, 52, 152, 53, 153, 54, 154.

`outlierwidth` Sets the border line width (in px) of the outlier sample points. Number in the interval [0, Inf].

`fig` A `go_figure` object.

`data` A data frame in which formula properties (e.g. `x = ~col`, `marker_size = ~col`) are evaluated. Default: the data given to `go_figure()`. Only the evaluated values are added to the trace. A `crosstalk::SharedData` works the same way, and links the trace to its crosstalk group (see `highlight()`).

`row` Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

col — Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

secondary_y — If TRUE, use the secondary y axis of the subplot.

Details

go_box() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_box() creates an object and adds it to a figure.

Nested objects documented here:

go_box_hoverlabel() box.hoverlabel

go_box_hoverlabel_font() box.hoverlabel.font: Sets the font used in hover labels.

go_box_legendgrouptitle() box.legendgrouptitle

go_box_legendgrouptitle_font() box.legendgrouptitle.font: Sets this legend group's title font.

go_box_line() box.line

go_box_marker() box.marker

go_box_marker_line() box.marker.line

go_box_selected() box.selected

go_box_selected_marker() box.selected.marker

go_box_unselected() box.unselected

go_box_unselected_marker() box.unselected.marker

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_box(x = 1:3, y = c(2, 1, 3), marker_color = "red")
```

go_candlestick *Candlestick traces*

Description

The candlestick is a style of financial chart describing open, high, low and close for a given ‘x‘ coordinate (most likely time). The boxes represent the spread between the ‘open‘ and ‘close‘ values and the lines represent the spread between the ‘low‘ and ‘high‘ values Sample points where the close value is higher (lower) than the open value are called increasing (decreasing). By default, increasing candles are drawn in green whereas decreasing are drawn in red.

Usage

```
go_candlestick(
  arg = NULL,
  ...,
  close = NULL,
  customdata = NULL,
  decreasing = NULL,
  high = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  increasing = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  low = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  open = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  whiskerwidth = NULL,
  x = NULL,
  xaxis = NULL,
```

```
  xcalendar = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  zorder = NULL,
  skip_invalid = FALSE
)

go_candlestick_decreasing(
  arg = NULL,
  ...,
  fillcolor = NULL,
  line = NULL,
  skip_invalid = FALSE
)

go_candlestick_decreasing_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_candlestick_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  split = NULL,
  skip_invalid = FALSE
)

go_candlestick_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
```

```
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_candlestick_increasing(
  arg = NULL,
  ...,
  fillcolor = NULL,
  line = NULL,
  skip_invalid = FALSE
)

go_candlestick_increasing_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_candlestick_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_candlestick_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_candlestick_line(arg = NULL, ..., width = NULL, skip_invalid = FALSE)
```

```
add_candlestick(fig, ...)

## Default S3 method:
add_candlestick(
  fig,
  ...,
  data = NULL,
  close = NULL,
  customdata = NULL,
  decreasing = NULL,
  high = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  increasing = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  low = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  open = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  whiskerwidth = NULL,
  x = NULL,
  xaxis = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  yaxis = NULL,
  yhoverformat = NULL,
  zorder = NULL,
  row = NULL,
  col = NULL,
```

```
  secondary_y = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
close	Sets the close values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
decreasing	Named list or `go_candlestick_decreasing()` object.
high	Sets the high values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
hoverinfo	Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of `"x"`, `"y"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.
hoverlabel	Named list or `go_candlestick_hoverlabel()` object.
hovertemplate	Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variables 'open', 'high', 'low' and 'close'. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide

the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

`hovertemplatefallback`
: Fallback string that’s displayed when a variable referenced in a template is missing. If the boolean value ’false’ is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext`
: Same as ‘text‘. String, or a vector of these.

`ids`
: Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`increasing`
: Named list or `go_candlestick_increasing()` object.

`legend`
: Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup`
: Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_candlestick_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line`
: `go_candlestick()`: Named list or `go_candlestick_line()` object.

`go_candlestick_decreasing()`: Named list or `go_candlestick_decreasing_line()` object.

`go_candlestick_increasing()`: Named list or `go_candlestick_increasing_line()` object.

`low`
: Sets the low values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name`
: Sets the trace name. The trace name appears as the legend item and on hover. String.

`opacity` Sets the opacity of the trace. Number in the interval [0, 1].

`open` Sets the open values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the 'unselected' are turned on for all points, whereas, any other non-array values means no selection all where the 'selected' and 'unselected' styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`text` `go_candlestick()`: Sets hover text elements associated with each sample point. If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to this trace's sample points. String, or a vector of these.

`go_candlestick_legendgrouptitle()`: Sets the title of the legend group. String.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as 'name' and 'colorbar.title'. Defaults to 'layout.uirevision'. Note that other user-driven trace attribute changes are controlled by 'layout' attributes: 'trace.visible' is controlled by 'layout.legend.uirevision', 'selectedpoints' is controlled by 'layout.selectionrevision', and 'colorbar.(xly)' (accessible with 'config: {editable: true}') is controlled by 'layout.editrevision'. Trace changes are tracked by 'uid', which only falls back on trace index if no 'uid' is provided. So if your app can add/remove traces before the end of the 'data' array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a 'uid' that stays with it as it moves. Any value.

`visible` Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`whiskerwidth` Sets the width of the whiskers relative to the box width. For example, with 1, the whiskers are as wide as the box(es). Number in the interval [0, 1].

`x` Sets the x coordinates. If absent, linear coordinate will be generated. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`xaxis` Sets a reference between this trace's x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to 'layout.xaxis'. If "x2", the x coordinates refer to 'layout.xaxis2', and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...

`xcalendar` Sets the calendar system to use with 'x' date data. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

xhoverformat Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

xperiod Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the x axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

xperiod0 Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the x0 axis. When ‘x0period‘ is round number of weeks, the ‘x0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

xperiodalignment
Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the x axis. One of `"start"`, `"middle"`, `"end"`.

yaxis Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...

yhoverformat Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

zorder Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher ‘zorder‘ appear in front of those with lower ‘zorder‘. Integer.

skip_invalid If `TRUE`, silently drop invalid properties instead of raising an error.

fillcolor Sets the fill color. Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name.

color `go_candlestick_decreasing_line()`,`go_candlestick_increasing_line()`: Sets the color of line bounding the box(es). Color: a CSS color string or R color name.

`go_candlestick_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

`go_candlestick_legendgrouptitle_font()`: Color: a CSS color string or R color name.

width
: go_candlestick_decreasing_line(),go_candlestick_increasing_line(): Sets the width (in px) of line bounding the box(es). Number in the interval [0, Inf].

  go_candlestick_line(): Sets the width (in px) of line bounding the box(es). Note that this style setting can also be set per direction via 'increasing.line.width' and 'decreasing.line.width'. Number in the interval [0, Inf].

align
: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.

bgcolor
: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

bordercolor
: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

font
: go_candlestick_hoverlabel(): Sets the font used in hover labels. Named list or go_candlestick_hoverlabel_font() object.

  go_candlestick_legendgrouptitle(): Sets this legend group's title font. Named list or go_candlestick_legendgrouptitle_font() object.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

split
: Show hover information (open, close, high, low) in separate labels, rather than a single unified label. Default: FALSE. When set to TRUE, 'hovertemplate' is ignored. Logical.

family
: go_candlestick_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

  go_candlestick_legendgrouptitle_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: go_candlestick_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

  go_candlestick_legendgrouptitle_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow
: go_candlestick_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

  go_candlestick_legendgrouptitle_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: go_candlestick_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

  go_candlestick_legendgrouptitle_font(): Number in the interval [1, Inf].

style
: go_candlestick_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

  go_candlestick_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: go_candlestick_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

  go_candlestick_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: go_candlestick_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

  go_candlestick_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: go_candlestick_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

  go_candlestick_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

fig
: A go_figure object.

data
: A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).

row
: Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.

col
: Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

secondary_y
: If TRUE, use the secondary y axis of the subplot.

Details

go_candlestick() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_candlestick() creates an object and adds it to a figure.

Nested objects documented here:

go_candlestick_decreasing() candlestick.decreasing

go_candlestick_decreasing_line() candlestick.decreasing.line

go_candlestick_hoverlabel() candlestick.hoverlabel

go_candlestick_hoverlabel_font() candlestick.hoverlabel.font: Sets the font used in hover labels.

go_candlestick_increasing() candlestick.increasing

go_candlestick_increasing_line() candlestick.increasing.line

go_candlestick_legendgrouptitle() candlestick.legendgrouptitle

go_candlestick_legendgrouptitle_font() candlestick.legendgrouptitle.font: Sets this legend group's title font.

go_candlestick_line() candlestick.line

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_candlestick()
```

---

go_carpet *Carpet traces*

Description

The data describing carpet axis layout is set in ‘y‘ and (optionally) also ‘x‘. If only ‘y‘ is present, ‘x‘ the plot is interpreted as a cheater plot and is filled in using the ‘y‘ values. ‘x‘ and ‘y‘ may either be 2D arrays matching with each dimension matching that of ‘a‘ and ‘b‘, or they may be 1D arrays with total length equal to that of ‘a‘ and ‘b‘.

Usage

```
go_carpet(
  arg = NULL,
  ...,
  a = NULL,
  a0 = NULL,
  aaxis = NULL,
  b = NULL,
  b0 = NULL,
  baxis = NULL,
  carpet = NULL,
  cheaterslope = NULL,
  color = NULL,
  customdata = NULL,
  da = NULL,
  db = NULL,
  font = NULL,
  ids = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  xaxis = NULL,
  y = NULL,
  yaxis = NULL,
  zorder = NULL,
  skip_invalid = FALSE
)

go_carpet_aaxis(
  arg = NULL,
  ...,
  arraydtick = NULL,
  arraytick0 = NULL,
  autorange = NULL,
  autotypenumbers = NULL,
  categoryarray = NULL,
  categoryorder = NULL,
  cheatertype = NULL,
  color = NULL,
  dtick = NULL,
```

```
    endline = NULL,
    endlinecolor = NULL,
    endlinewidth = NULL,
    exponentformat = NULL,
    fixedrange = NULL,
    gridcolor = NULL,
    griddash = NULL,
    gridwidth = NULL,
    labelalias = NULL,
    labelpadding = NULL,
    labelprefix = NULL,
    labelsuffix = NULL,
    linecolor = NULL,
    linewidth = NULL,
    minexponent = NULL,
    minorgridcolor = NULL,
    minorgridcount = NULL,
    minorgriddash = NULL,
    minorgridwidth = NULL,
    nticks = NULL,
    range = NULL,
    rangemode = NULL,
    separatethousands = NULL,
    showexponent = NULL,
    showgrid = NULL,
    showline = NULL,
    showticklabels = NULL,
    showtickprefix = NULL,
    showticksuffix = NULL,
    smoothing = NULL,
    startline = NULL,
    startlinecolor = NULL,
    startlinewidth = NULL,
    tick0 = NULL,
    tickangle = NULL,
    tickfont = NULL,
    tickformat = NULL,
    tickformatstops = NULL,
    tickmode = NULL,
    tickprefix = NULL,
    ticksuffix = NULL,
    ticktext = NULL,
    tickvals = NULL,
    title = NULL,
    type = NULL,
    skip_invalid = FALSE
  )
```

```
  go_carpet_aaxis_tickfont(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_carpet_aaxis_tickformatstop(
    arg = NULL,
    ...,
    dtickrange = NULL,
    enabled = NULL,
    name = NULL,
    templateitemname = NULL,
    value = NULL,
    skip_invalid = FALSE
  )

  go_carpet_aaxis_title(
    arg = NULL,
    ...,
    font = NULL,
    offset = NULL,
    text = NULL,
    skip_invalid = FALSE
  )

  go_carpet_aaxis_title_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
```

```
)

go_carpet_baxis(
  arg = NULL,
  ...,
  arraydtick = NULL,
  arraytick0 = NULL,
  autorange = NULL,
  autotypenumbers = NULL,
  categoryarray = NULL,
  categoryorder = NULL,
  cheatertype = NULL,
  color = NULL,
  dtick = NULL,
  endline = NULL,
  endlinecolor = NULL,
  endlinewidth = NULL,
  exponentformat = NULL,
  fixedrange = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
  labelalias = NULL,
  labelpadding = NULL,
  labelprefix = NULL,
  labelsuffix = NULL,
  linecolor = NULL,
  linewidth = NULL,
  minexponent = NULL,
  minorgridcolor = NULL,
  minorgridcount = NULL,
  minorgriddash = NULL,
  minorgridwidth = NULL,
  nticks = NULL,
  range = NULL,
  rangemode = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showgrid = NULL,
  showline = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  smoothing = NULL,
  startline = NULL,
  startlinecolor = NULL,
  startlinewidth = NULL,
  tick0 = NULL,
```

```
  tickangle = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  title = NULL,
  type = NULL,
  skip_invalid = FALSE
)

go_carpet_baxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_carpet_baxis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_carpet_baxis_title(
  arg = NULL,
  ...,
  font = NULL,
  offset = NULL,
  text = NULL,
  skip_invalid = FALSE
)
```

```
  go_carpet_baxis_title_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_carpet_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
    textcase = NULL,
    variant = NULL,
    weight = NULL,
    skip_invalid = FALSE
  )

  go_carpet_legendgrouptitle(
    arg = NULL,
    ...,
    font = NULL,
    text = NULL,
    skip_invalid = FALSE
  )

  go_carpet_legendgrouptitle_font(
    arg = NULL,
    ...,
    color = NULL,
    family = NULL,
    lineposition = NULL,
    shadow = NULL,
    size = NULL,
    style = NULL,
```

```
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

add_carpet(fig, ...)

## Default S3 method:
add_carpet(
  fig,
  ...,
  data = NULL,
  a = NULL,
  a0 = NULL,
  aaxis = NULL,
  b = NULL,
  b0 = NULL,
  baxis = NULL,
  carpet = NULL,
  cheaterslope = NULL,
  color = NULL,
  customdata = NULL,
  da = NULL,
  db = NULL,
  font = NULL,
  ids = NULL,
  legend = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  xaxis = NULL,
  y = NULL,
  yaxis = NULL,
  zorder = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
a	An array containing values of the first parameter value Data array: a vector, factor, Date/POSIXct vector, matrix or list.
a0	Alternate to 'a'. Builds a linear space of a coordinates. Use with 'da' where 'a0' is the starting coordinate and 'da' the step. Number.
aaxis	Named list or go_carpet_aaxis() object.
b	A two dimensional array of y coordinates at each carpet point. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
b0	Alternate to 'b'. Builds a linear space of a coordinates. Use with 'db' where 'b0' is the starting coordinate and 'db' the step. Number.
baxis	Named list or go_carpet_baxis() object.
carpet	An identifier for this carpet, so that 'scattercarpet' and 'contourcarpet' traces can specify a carpet plot on which they lie String.
cheaterslope	The shift applied to each successive row of data in creating a cheater plot. Only used if 'x' is been omitted. Number.
color	go_carpet(), go_carpet_aaxis(), go_carpet_baxis(): Sets default for all colors associated with this axis all at once: line, font, tick, and grid colors. Grid color is lightened by blending this with the plot background Individual pieces can override this. Color: a CSS color string or R color name. go_carpet_aaxis_tickfont(), go_carpet_aaxis_title_font(), go_carpet_baxis_tickfont(), go_carpet_baxis_title_font(), go_carpet_font(), go_carpet_legendgrouptitle_font(): Color: a CSS color string or R color name.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
da	Sets the a coordinate step. See 'a0' for more info. Number.
db	Sets the b coordinate step. See 'b0' for more info. Number.
font	go_carpet(): The default font used for axis & tick labels on this carpet Named list or go_carpet_font() object. go_carpet_aaxis_title(): Sets this axis' title font. Named list or go_carpet_aaxis_title_font() object. go_carpet_baxis_title(): Sets this axis' title font. Named list or go_carpet_baxis_title_font() object. go_carpet_legendgrouptitle(): Sets this legend group's title font. Named list or go_carpet_legendgrouptitle_font() object.
ids	Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgrouptitle`
Named list or `go_carpet_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_carpet()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_carpet_aaxis_tickformatstop()`,`go_carpet_baxis_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity` Sets the opacity of the trace. Number in the interval [0, 1].

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

visible — Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.

x — A two dimensional array of x coordinates at each carpet point. If omitted, the plot is a cheater plot and the xaxis is hidden by default. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

xaxis — Sets a reference between this trace's x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to 'layout.xaxis'. If "x2", the x coordinates refer to 'layout.xaxis2', and so on. Subplot id: "x", "x2", "x3", ...

y — A two dimensional array of y coordinates at each carpet point. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

yaxis — Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to 'layout.yaxis'. If "y2", the y coordinates refer to 'layout.yaxis2', and so on. Subplot id: "y", "y2", "y3", ...

zorder — Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher 'zorder' appear in front of those with lower 'zorder'. Integer.

skip_invalid — If TRUE, silently drop invalid properties instead of raising an error.

arraydtick — The stride between grid lines along the axis Integer in the interval [1, Inf].

arraytick0 — The starting index of grid lines along the axis Integer in the interval [0, Inf].

autorange — Determines whether or not the range of this axis is computed in relation to the input data. See 'rangemode' for more info. If 'range' is provided, then 'autorange' is set to FALSE. One of "reversed", TRUE, FALSE.

autotypenumbers — Using "strict" a numeric string in trace data is not converted to a number. Using "convert types" a numeric string in trace data may be treated as a number during automatic axis 'type' detection. Defaults to layout.autotypenumbers. One of "convert types", "strict".

categoryarray — Sets the order in which categories on this axis appear. Only has an effect if 'categoryorder' is set to "array". Used with 'categoryorder'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

categoryorder — Specifies the ordering logic for the case of categorical variables. By default, plotly uses "trace", which specifies the order that is present in the data supplied. Set 'categoryorder' to "category ascending" or "category descending" if order should be determined by the alphanumerical order of the category names. Set 'categoryorder' to "array" to derive the ordering from the attribute 'categoryarray'. If a category is not found in the 'categoryarray' array, the sorting behavior for that attribute will be identical to the "trace" mode. The unspecified categories will follow the categories in 'categoryarray'. One of "trace", "category ascending", "category descending", "array".

cheatertype — One of "index", "value".

dtick — The stride between grid lines along the axis Number in the interval [0, Inf].

endline — Determines whether or not a line is drawn at along the final value of this axis. If TRUE, the end line is drawn on top of the grid lines. Logical.

endlinecolor Sets the line color of the end line. Color: a CSS color string or R color name.

endlinewidth Sets the width (in px) of the end line. Number.

exponentformat Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

fixedrange Determines whether or not this axis is zoom-able. If true, then zoom is disabled. Logical.

gridcolor Sets the axis line color. Color: a CSS color string or R color name.

griddash Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

gridwidth Sets the width (in px) of the axis line. Number in the interval [0, Inf].

labelalias Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

labelpadding Extra padding between label and the axis Integer.

labelprefix Sets a axis label prefix. String.

labelsuffix Sets a axis label suffix. String.

linecolor Sets the axis line color. Color: a CSS color string or R color name.

linewidth Sets the width (in px) of the axis line. Number in the interval [0, Inf].

minexponent Hide SI prefix for 10^n if |n| is below this number Number in the interval [0, Inf].

minorgridcolor Sets the color of the grid lines. Color: a CSS color string or R color name.

minorgridcount Sets the number of minor grid ticks per major grid tick Integer in the interval [0, Inf].

minorgriddash Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

minorgridwidth Sets the width (in px) of the grid lines. Number in the interval [0, Inf].

nticks Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

range Sets the range of this axis. If the axis 'type' is "log", then you must take the log of your desired range (e.g. to set the range from 1 to 100, set the range from 0 to 2). If the axis 'type' is "date", it should be date strings, like date data, though Date objects and unix milliseconds will be accepted and converted to strings. If

the axis ‘type‘ is "category", it should be numbers, using the scale where each category is assigned a serial number from zero in the order it appears. Vector or list of 2 element(s).

`rangemode` If "normal", the range is computed in relation to the extrema of the input data. If "tozero", the range extends to 0, regardless of the input data If "nonnegative", the range is non-negative, regardless of the input data. One of `"normal"`, `"tozero"`, `"nonnegative"`.

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showgrid` Determines whether or not grid lines are drawn. If TRUE, the grid lines are drawn at every tick mark. Logical.

`showline` Determines whether or not a line bounding this axis is drawn. Logical.

`showticklabels` Determines whether axis labels are drawn on the low side, the high side, both, or neither side of the axis. One of `"start"`, `"end"`, `"both"`, `"none"`.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`smoothing` Number in the interval [0, 1.3].

`startline` Determines whether or not a line is drawn at along the starting value of this axis. If TRUE, the start line is drawn on top of the grid lines. Logical.

`startlinecolor` Sets the line color of the start line. Color: a CSS color string or R color name.

`startlinewidth` Sets the width (in px) of the start line. Number.

`tick0` The starting index of grid lines along the axis Number in the interval [0, Inf].

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickfont` `go_carpet_aaxis()`: Sets the tick font. Named list or `go_carpet_aaxis_tickfont()` object.

`go_carpet_baxis()`: Sets the tick font. Named list or `go_carpet_baxis_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops
: go_carpet_aaxis(): List of named lists or go_carpet_aaxis_tickformatstop() objects.
go_carpet_baxis(): List of named lists or go_carpet_baxis_tickformatstop() objects.

tickmode
: One of "linear", "array".

tickprefix
: Sets a tick label prefix. String.

ticksuffix
: Sets a tick label suffix. String.

ticktext
: Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals
: Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

title
: go_carpet_aaxis(): String (the title text), named list or go_carpet_aaxis_title() object.
go_carpet_baxis(): String (the title text), named list or go_carpet_baxis_title() object.

type
: Sets the axis type. By default, plotly attempts to determined the axis type by looking into the data of the traces that referenced the axis in question. One of "-", "linear", "date", "category".

family
: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow
: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: Number in the interval [1, Inf].

style
: Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant
: Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight
: Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

dtickrange
: range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

offset — An additional amount by which to offset the title from the tick labels, given in pixels. Number.

text — go_carpet_aaxis_title(), go_carpet_baxis_title(): Sets the title of this axis. String.

go_carpet_legendgrouptitle(): Sets the title of the legend group. String.

fig — A go_figure object.

data — A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).

row — Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.

col — Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

secondary_y — If TRUE, use the secondary y axis of the subplot.

Details

go_carpet() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_carpet() creates an object and adds it to a figure.

Nested objects documented here:

go_carpet_aaxis() carpet.aaxis

go_carpet_aaxis_tickfont() carpet.aaxis.tickfont: Sets the tick font.

go_carpet_aaxis_tickformatstop() carpet.aaxis.tickformatstop

go_carpet_aaxis_title() carpet.aaxis.title

go_carpet_aaxis_title_font() carpet.aaxis.title.font: Sets this axis' title font.

go_carpet_baxis() carpet.baxis

go_carpet_baxis_tickfont() carpet.baxis.tickfont: Sets the tick font.

go_carpet_baxis_tickformatstop() carpet.baxis.tickformatstop

go_carpet_baxis_title() carpet.baxis.title

go_carpet_baxis_title_font() carpet.baxis.title.font: Sets this axis' title font.

`go_carpet_font()` `carpet.font`: The default font used for axis & tick labels on this carpet

`go_carpet_legendgrouptitle()` `carpet.legendgrouptitle`

`go_carpet_legendgrouptitle_font()` `carpet.legendgrouptitle.font`: Sets this legend group's title font.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_carpet(x = 1:3, y = c(2, 1, 3))
```

---

go_choropleth *Choropleth traces*

Description

The data that describes the choropleth value-to-color mapping is set in 'z'. The geographic locations corresponding to each value in 'z' are set in 'locations'.

Usage

```
go_choropleth(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  customdata = NULL,
  featureidkey = NULL,
  geo = NULL,
  geojson = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
```

```
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  locationmode = NULL,
  locations = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  reversescale = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  showscale = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  z = NULL,
  zauto = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  skip_invalid = FALSE
)

go_choropleth_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
```

```
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_choropleth_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_choropleth_colorbar_tickformatstop(
  arg = NULL,
  ...,
```

```
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_choropleth_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_choropleth_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_choropleth_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_choropleth_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
```

```
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_choropleth_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_choropleth_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_choropleth_marker(
  arg = NULL,
  ...,
  line = NULL,
  opacity = NULL,
  skip_invalid = FALSE
)

go_choropleth_marker_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
```

```
)

go_choropleth_selected(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

go_choropleth_selected_marker(
  arg = NULL,
  ...,
  opacity = NULL,
  skip_invalid = FALSE
)

go_choropleth_unselected(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

go_choropleth_unselected_marker(
  arg = NULL,
  ...,
  opacity = NULL,
  skip_invalid = FALSE
)

add_choropleth(fig, ...)

## Default S3 method:
add_choropleth(
  fig,
  ...,
  data = NULL,
  autocolorscale = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  customdata = NULL,
  featureidkey = NULL,
  geo = NULL,
  geojson = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  locationmode = NULL,
  locations = NULL,
```

```
  marker = NULL,
  meta = NULL,
  name = NULL,
  reversescale = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  showscale = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
  z = NULL,
  zauto = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)
```

Arguments

`arg`	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
`...`	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
`autocolorscale`	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
`coloraxis`	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...
`colorbar`	Named list or `go_choropleth_colorbar()` object.
`colorscale`	Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘zmin‘ and ‘zmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi

Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).

customdata Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

featureidkey Sets the key in GeoJSON features which is used as id to match the items included in the 'locations' array. Only has an effect when 'geojson' is set. Support nested property, for example "properties.name". String.

geo Sets a reference between this trace's geospatial coordinates and a geographic map. If "geo" (the default value), the geospatial coordinates refer to 'layout.geo'. If "geo2", the geospatial coordinates refer to 'layout.geo2', and so on. Subplot id: "geo", "geo2", "geo3", ...

geojson Sets optional GeoJSON data associated with this trace. If not given, the features on the base map are used. It can be set as a valid GeoJSON object or as a URL string. Note that we only accept GeoJSONs of type "FeatureCollection" or "Feature" with geometries of type "Polygon" or "MultiPolygon". Any value.

hoverinfo Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of "location", "z", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.

hoverlabel Named list or go_choropleth_hoverlabel() object.

hovertemplate Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

hovertemplatefallback Fallback string that's displayed when a variable referenced in a template is miss-

ing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Same as 'text'. String, or a vector of these.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
Named list or `go_choropleth_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`locationmode` Determines the set of locations used to match entries in 'locations' to regions on the map. Values "ISO-3", "USA-states", "country names" correspond to features on the base map and value "geojson-id" corresponds to features from a custom GeoJSON linked to the 'geojson' attribute. "USA-states" accepts both two-letter abbreviations (e.g. "CA") and full state names (e.g. "California"). One of `"ISO-3"`, `"USA-states"`, `"country names"`, `"geojson-id"`.

`locations` Sets the coordinates via location IDs or names. See 'locationmode' for more info. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`marker` `go_choropleth()`: Named list or `go_choropleth_marker()` object.

`go_choropleth_selected()`: Named list or `go_choropleth_selected_marker()` object.

`go_choropleth_unselected()`: Named list or `go_choropleth_unselected_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

name	go_choropleth(): Sets the trace name. The trace name appears as the legend item and on hover. String. go_choropleth_colorbar_tickformatstop(): When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.
reversescale	Reverses the color mapping if true. If true, ‘zmin‘ will correspond to the last color in the array and ‘zmax‘ will correspond to the first color. Logical.
selected	Named list or go_choropleth_selected() object.
selectedpoints	Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.
showlegend	Determines whether or not an item corresponding to this trace is shown in the legend. Logical.
showscale	Determines whether or not a colorbar is displayed for this trace. Logical.
text	go_choropleth(): Sets the text elements associated with each location. String, or a vector of these. go_choropleth_colorbar_title(): Sets the title of the color bar. String. go_choropleth_legendgrouptitle(): Sets the title of the legend group. String.
uid	Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.
uirevision	Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.
unselected	Named list or go_choropleth_unselected() object.
visible	Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.
z	Sets the color values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
zauto	Determines whether or not the color domain is computed with respect to the input data (here in ‘z‘) or the bounds set in ‘zmin‘ and ‘zmax‘ Defaults to ‘false‘ when ‘zmin‘ and ‘zmax‘ are set by the user. Logical.

zmax	Sets the upper bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmin‘ must be set as well. Number.
zmid	Sets the mid-point of the color domain by scaling ‘zmin‘ and/or ‘zmax‘ to be equidistant to this point. Value should have the same units as in ‘z‘. Has no effect when ‘zauto‘ is ‘false‘. Number.
zmin	Sets the lower bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmax‘ must be set as well. Number.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
bgcolor	go_choropleth_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name. go_choropleth_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.
bordercolor	go_choropleth_colorbar(): Sets the axis line color. Color: a CSS color string or R color name. go_choropleth_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.
borderwidth	Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].
dtick	Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.
exponentformat	Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".
labelalias	Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.
len	Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].
lenmode	Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of "fraction", "pixels".
minexponent	Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

nticks Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

orientation Sets the orientation of the colorbar. One of "h", "v".

outlinecolor Sets the axis line color. Color: a CSS color string or R color name.

outlinewidth Sets the width (in px) of the axis line. Number in the interval [0, Inf].

separatethousands
If "true", even 4-digit integers are separated Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".

showticksuffix Same as ‘showtickprefix‘ but for tick suffixes. One of "all", "first", "last", "none".

thickness Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of "fraction", "pixels".

tick0 Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar’s tick label font Named list or go_choropleth_colorbar_tickfont() object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops
List of named lists or go_choropleth_colorbar_tickformatstop() objects.

ticklabeloverflow
: Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

ticklabelposition
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

ticklabelstep
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

ticklen
: Sets the tick length (in px). Number in the interval [0, Inf].

tickmode
: Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

tickprefix
: Sets a tick label prefix. String.

ticks
: Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

ticksuffix
: Sets a tick label suffix. String.

ticktext
: Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals
: Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth
: Sets the tick width (in px). Number in the interval [0, Inf].

title
: String (the title text), named list or `go_choropleth_colorbar_title()` object.

x
: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

xanchor
: Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

xpad — Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

y — Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

yanchor — Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

color — `go_choropleth_colorbar_tickfont()`, `go_choropleth_colorbar_title_font()`, `go_choropleth_legendgrouptitle_font()`: Color: a CSS color string or R color name.

`go_choropleth_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

`go_choropleth_marker_line()`: Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name, or a vector of these.

family — `go_choropleth_colorbar_tickfont()`, `go_choropleth_colorbar_title_font()`, `go_choropleth_legendgrouptitle_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String.

`go_choropleth_hoverlabel_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String, or a vector of these.

lineposition — `go_choropleth_colorbar_tickfont()`, `go_choropleth_colorbar_title_font()`, `go_choropleth_legendgrouptitle_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

go_choropleth_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

shadow
go_choropleth_colorbar_tickfont(), go_choropleth_colorbar_title_font(), go_choropleth_legendgrouptitle_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

go_choropleth_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size
go_choropleth_colorbar_tickfont(), go_choropleth_colorbar_title_font(), go_choropleth_legendgrouptitle_font(): Number in the interval [1, Inf].

go_choropleth_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

style
go_choropleth_colorbar_tickfont(), go_choropleth_colorbar_title_font(), go_choropleth_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

go_choropleth_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

textcase
go_choropleth_colorbar_tickfont(), go_choropleth_colorbar_title_font(), go_choropleth_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

go_choropleth_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant
go_choropleth_colorbar_tickfont(), go_choropleth_colorbar_title_font(), go_choropleth_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

go_choropleth_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight
go_choropleth_colorbar_tickfont(), go_choropleth_colorbar_title_font(), go_choropleth_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

go_choropleth_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange
range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled
: Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

templateitemname
: Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

value
: string - dtickformat for described zoom level, the same as "tickformat" String.

font
: `go_choropleth_colorbar_title()`: Sets this color bar's title font. Named list or `go_choropleth_colorbar_title_font()` object.

`go_choropleth_hoverlabel()`: Sets the font used in hover labels. Named list or `go_choropleth_hoverlabel_font()` object.

`go_choropleth_legendgrouptitle()`: Sets this legend group's title font. Named list or `go_choropleth_legendgrouptitle_font()` object.

side
: Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of `"right"`, `"top"`, `"bottom"`.

align
: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`, or a vector of these.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow
: Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

line
: Named list or `go_choropleth_marker_line()` object.

opacity
: `go_choropleth_marker()`: Sets the opacity of the locations. Number in the interval [0, 1], or a vector of these.

`go_choropleth_selected_marker()`: Sets the marker opacity of selected points. Number in the interval [0, 1].

`go_choropleth_unselected_marker()`: Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].

width
: Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.

fig
: A `go_figure` object.

data
: A data frame in which formula properties (e.g. `x = ~col`, `marker_size = ~col`) are evaluated. Default: the data given to `go_figure()`. Only the evaluated values are added to the trace. A `crosstalk::SharedData` works the same way, and links the trace to its crosstalk group (see `highlight()`).

row — Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.

col — Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

Details

go_choropleth() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_choropleth() creates an object and adds it to a figure.

Nested objects documented here:

go_choropleth_colorbar() choropleth.colorbar

go_choropleth_colorbar_tickfont() choropleth.colorbar.tickfont: Sets the color bar's tick label font

go_choropleth_colorbar_tickformatstop() choropleth.colorbar.tickformatstop

go_choropleth_colorbar_title() choropleth.colorbar.title

go_choropleth_colorbar_title_font() choropleth.colorbar.title.font: Sets this color bar's title font.

go_choropleth_hoverlabel() choropleth.hoverlabel

go_choropleth_hoverlabel_font() choropleth.hoverlabel.font: Sets the font used in hover labels.

go_choropleth_legendgrouptitle() choropleth.legendgrouptitle

go_choropleth_legendgrouptitle_font() choropleth.legendgrouptitle.font: Sets this legend group's title font.

go_choropleth_marker() choropleth.marker

go_choropleth_marker_line() choropleth.marker.line

go_choropleth_selected() choropleth.selected

go_choropleth_selected_marker() choropleth.selected.marker

go_choropleth_unselected() choropleth.unselected

go_choropleth_unselected_marker() choropleth.unselected.marker

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_choropleth()
```

go_choroplethmap *Choroplethmap traces*

Description

GeoJSON features to be filled are set in 'geojson' The data that describes the choropleth value-to-color mapping is set in 'locations' and 'z'.

Usage

```
go_choroplethmap(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  below = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  customdata = NULL,
  featureidkey = NULL,
  geojson = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  locations = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  reversescale = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  showscale = NULL,
  subplot = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
```

```
  z = NULL,
  zauto = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
```

```
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_colorbar_title_font(
```

```
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)
```

```
go_choroplethmap_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_marker(
  arg = NULL,
  ...,
  line = NULL,
  opacity = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_marker_line(
  arg = NULL,
  ...,
  color = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_selected(arg = NULL, ..., marker = NULL, skip_invalid = FALSE)

go_choroplethmap_selected_marker(
  arg = NULL,
  ...,
  opacity = NULL,
  skip_invalid = FALSE
)

go_choroplethmap_unselected(
  arg = NULL,
  ...,
  marker = NULL,
  skip_invalid = FALSE
)
```

```
go_choroplethmap_unselected_marker(
  arg = NULL,
  ...,
  opacity = NULL,
  skip_invalid = FALSE
)

add_choroplethmap(fig, ...)

## Default S3 method:
add_choroplethmap(
  fig,
  ...,
  data = NULL,
  autocolorscale = NULL,
  below = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  customdata = NULL,
  featureidkey = NULL,
  geojson = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  locations = NULL,
  marker = NULL,
  meta = NULL,
  name = NULL,
  reversescale = NULL,
  selected = NULL,
  selectedpoints = NULL,
  showlegend = NULL,
  showscale = NULL,
  subplot = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  unselected = NULL,
  visible = NULL,
```

```
  z = NULL,
  zauto = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
autocolorscale	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
below	Determines if the choropleth polygons will be inserted before the layer with the specified ID. By default, choroplethmap traces are placed above the water layers. If set to ”, the layer will be inserted above every existing layer. String.
coloraxis	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...
colorbar	Named list or `go_choroplethmap_colorbar()` object.
colorscale	Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘zmin‘ and ‘zmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
featureidkey	Sets the key in GeoJSON features which is used as id to match the items included in the ‘locations‘ array. Support nested property, for example "properties.name". String.

`geojson` Sets the GeoJSON data associated with this trace. It can be set as a valid GeoJSON object or as a URL string. Note that we only accept GeoJSONs of type "FeatureCollection" or "Feature" with geometries of type "Polygon" or "MultiPolygon". Any value.

`hoverinfo` Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of `"location"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_choroplethmap_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Finally, the template string has access to variable 'properties' Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Same as 'text'. String, or a vector of these.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle` Named list or `go_choroplethmap_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`locations` Sets which features found in "geojson" to plot using their feature ‘id‘ field. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`marker` `go_choroplethmap()`: Named list or `go_choroplethmap_marker()` object.

`go_choroplethmap_selected()`: Named list or `go_choroplethmap_selected_marker()` object.

`go_choroplethmap_unselected()`: Named list or `go_choroplethmap_unselected_marker()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name` `go_choroplethmap()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_choroplethmap_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`reversescale` Reverses the color mapping if true. If true, ‘zmin‘ will correspond to the last color in the array and ‘zmax‘ will correspond to the first color. Logical.

`selected` Named list or `go_choroplethmap_selected()` object.

`selectedpoints` Array containing integer indices of selected points. Has an effect only for traces that support selections. Note that an empty array means an empty selection where the ‘unselected‘ are turned on for all points, whereas, any other non-array values means no selection all where the ‘selected‘ and ‘unselected‘ styles have no effect. Any value.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Logical.

`subplot` Sets a reference between this trace’s data coordinates and a map subplot. If "map" (the default value), the data refer to ‘layout.map‘. If "map2", the data refer to ‘layout.map2‘, and so on. Subplot id: `"map"`, `"map2"`, `"map3"`, ...

text	go_choroplethmap(): Sets the text elements associated with each location. String, or a vector of these. go_choroplethmap_colorbar_title(): Sets the title of the color bar. String. go_choroplethmap_legendgrouptitle(): Sets the title of the legend group. String.
uid	Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.
uirevision	Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.
unselected	Named list or go_choroplethmap_unselected() object.
visible	Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.
z	Sets the color values. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
zauto	Determines whether or not the color domain is computed with respect to the input data (here in ‘z‘) or the bounds set in ‘zmin‘ and ‘zmax‘ Defaults to ‘false‘ when ‘zmin‘ and ‘zmax‘ are set by the user. Logical.
zmax	Sets the upper bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmin‘ must be set as well. Number.
zmid	Sets the mid-point of the color domain by scaling ‘zmin‘ and/or ‘zmax‘ to be equidistant to this point. Value should have the same units as in ‘z‘. Has no effect when ‘zauto‘ is ‘false‘. Number.
zmin	Sets the lower bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmax‘ must be set as well. Number.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
bgcolor	go_choroplethmap_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name. go_choroplethmap_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.
bordercolor	go_choroplethmap_colorbar(): Sets the axis line color. Color: a CSS color string or R color name. go_choroplethmap_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

borderwidth Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

dtick Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

exponentformat Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

labelalias Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

len Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

lenmode Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of "fraction", "pixels".

minexponent Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

nticks Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

orientation Sets the orientation of the colorbar. One of "h", "v".

outlinecolor Sets the axis line color. Color: a CSS color string or R color name.

outlinewidth Sets the width (in px) of the axis line. Number in the interval [0, Inf].

separatethousands If "true", even 4-digit integers are separated Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".

showticksuffix Same as ‘showtickprefix‘ but for tick suffixes. One of "all", "first", "last", "none".

thickness Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of "fraction", "pixels".

tick0 Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar's tick label font Named list or go_choroplethmap_colorbar_tickfont() object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or go_choroplethmap_colorbar_tickformatstop() objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelstep Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_choroplethmap_colorbar_title()` object.

`x` Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`xanchor` Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`y` Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

`yanchor` Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

color — `go_choroplethmap_colorbar_tickfont()`, `go_choroplethmap_colorbar_title_font()`, `go_choroplethmap_legendgrouptitle_font()`: Color: a CSS color string or R color name.

`go_choroplethmap_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

`go_choroplethmap_marker_line()`: Sets the marker.line color. It accepts either a specific color or an array of numbers that are mapped to the colorscale relative to the max and min values of the array or relative to ‘marker.line.cmin‘ and ‘marker.line.cmax‘ if set. Color: a CSS color string or R color name, or a vector of these.

family — `go_choroplethmap_colorbar_tickfont()`, `go_choroplethmap_colorbar_title_font()`, `go_choroplethmap_legendgrouptitle_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String.

`go_choroplethmap_hoverlabel_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String, or a vector of these.

lineposition — `go_choroplethmap_colorbar_tickfont()`, `go_choroplethmap_colorbar_title_font()`, `go_choroplethmap_legendgrouptitle_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

`go_choroplethmap_hoverlabel_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

shadow — `go_choroplethmap_colorbar_tickfont()`, `go_choroplethmap_colorbar_title_font()`, `go_choroplethmap_legendgrouptitle_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

`go_choroplethmap_hoverlabel_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size — `go_choroplethmap_colorbar_tickfont()`, `go_choroplethmap_colorbar_title_font()`, `go_choroplethmap_legendgrouptitle_font()`: Number in the interval [1, Inf].

go_choroplethmap_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

style — go_choroplethmap_colorbar_tickfont(), go_choroplethmap_colorbar_title_font(), go_choroplethmap_legendgrouptitle_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

go_choroplethmap_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

textcase — go_choroplethmap_colorbar_tickfont(), go_choroplethmap_colorbar_title_font(), go_choroplethmap_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

go_choroplethmap_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant — go_choroplethmap_colorbar_tickfont(), go_choroplethmap_colorbar_title_font(), go_choroplethmap_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

go_choroplethmap_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight — go_choroplethmap_colorbar_tickfont(), go_choroplethmap_colorbar_title_font(), go_choroplethmap_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

go_choroplethmap_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

font — go_choroplethmap_colorbar_title(): Sets this color bar's title font. Named list or go_choroplethmap_colorbar_title_font() object.

go_choroplethmap_hoverlabel(): Sets the font used in hover labels. Named list or go_choroplethmap_hoverlabel_font() object.

	go_choroplethmap_legendgrouptitle(): Sets this legend group's title font. Named list or go_choroplethmap_legendgrouptitle_font() object.
side	Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of "right", "top", "bottom".
align	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.
namelength	Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.
showarrow	Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.
line	Named list or go_choroplethmap_marker_line() object.
opacity	go_choroplethmap_marker(): Sets the opacity of the locations. Number in the interval [0, 1], or a vector of these. go_choroplethmap_selected_marker(): Sets the marker opacity of selected points. Number in the interval [0, 1]. go_choroplethmap_unselected_marker(): Sets the marker opacity of unselected points, applied only when a selection exists. Number in the interval [0, 1].
width	Sets the width (in px) of the lines bounding the marker points. Number in the interval [0, Inf], or a vector of these.
fig	A go_figure object.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
row	Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

Details

go_choroplethmap() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_choroplethmap() creates an object and adds it to a figure.

Nested objects documented here:

go_choroplethmap_colorbar() choroplethmap.colorbar

`go_choroplethmap_colorbar_tickfont()` `choroplethmap.colorbar.tickfont`: Sets the color bar's tick label font

`go_choroplethmap_colorbar_tickformatstop()` `choroplethmap.colorbar.tickformatstop`

`go_choroplethmap_colorbar_title()` `choroplethmap.colorbar.title`

`go_choroplethmap_colorbar_title_font()` `choroplethmap.colorbar.title.font`: Sets this color bar's title font.

`go_choroplethmap_hoverlabel()` `choroplethmap.hoverlabel`

`go_choroplethmap_hoverlabel_font()` `choroplethmap.hoverlabel.font`: Sets the font used in hover labels.

`go_choroplethmap_legendgrouptitle()` `choroplethmap.legendgrouptitle`

`go_choroplethmap_legendgrouptitle_font()` `choroplethmap.legendgrouptitle.font`: Sets this legend group's title font.

`go_choroplethmap_marker()` `choroplethmap.marker`

`go_choroplethmap_marker_line()` `choroplethmap.marker.line`

`go_choroplethmap_selected()` `choroplethmap.selected`

`go_choroplethmap_selected_marker()` `choroplethmap.selected.marker`

`go_choroplethmap_unselected()` `choroplethmap.unselected`

`go_choroplethmap_unselected_marker()` `choroplethmap.unselected.marker`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_choroplethmap()
```

---

go_cone *Cone traces*

Description

Use cone traces to visualize vector fields. Specify a vector field using 6 1D arrays, 3 position arrays 'x', 'y' and 'z' and 3 vector component arrays 'u', 'v', 'w'. The cones are drawn exactly at the positions given by 'x', 'y' and 'z'.

Usage

```
go_cone(
  arg = NULL,
  ...,
  anchor = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  customdata = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lighting = NULL,
  lightposition = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  reversescale = NULL,
  scene = NULL,
  showlegend = NULL,
  showscale = NULL,
  sizemode = NULL,
  sizeref = NULL,
  text = NULL,
  u = NULL,
  uhoverformat = NULL,
  uid = NULL,
  uirevision = NULL,
  v = NULL,
  vhoverformat = NULL,
  visible = NULL,
  w = NULL,
  whoverformat = NULL,
  x = NULL,
  xhoverformat = NULL,
```

```
  y = NULL,
  yhoverformat = NULL,
  z = NULL,
  zhoverformat = NULL,
  skip_invalid = FALSE
)

go_cone_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
```

```
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_cone_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_cone_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_cone_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_cone_colorbar_title_font(
  arg = NULL,
```

```
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_cone_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_cone_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_cone_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)
```

```
go_cone_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_cone_lighting(
  arg = NULL,
  ...,
  ambient = NULL,
  diffuse = NULL,
  facenormalsepsilon = NULL,
  fresnel = NULL,
  roughness = NULL,
  specular = NULL,
  vertexnormalsepsilon = NULL,
  skip_invalid = FALSE
)

go_cone_lightposition(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

add_cone(fig, ...)

## Default S3 method:
add_cone(
  fig,
  ...,
  data = NULL,
  anchor = NULL,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
```

```
  cmid = NULL,
  cmin = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  customdata = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lighting = NULL,
  lightposition = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  reversescale = NULL,
  scene = NULL,
  showlegend = NULL,
  showscale = NULL,
  sizemode = NULL,
  sizeref = NULL,
  text = NULL,
  u = NULL,
  uhoverformat = NULL,
  uid = NULL,
  uirevision = NULL,
  v = NULL,
  vhoverformat = NULL,
  visible = NULL,
  w = NULL,
  whoverformat = NULL,
  x = NULL,
  xhoverformat = NULL,
  y = NULL,
  yhoverformat = NULL,
  z = NULL,
  zhoverformat = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
anchor	Sets the cones' anchor with respect to their x/y/z positions. Note that "cm" denote the cone's center of mass which corresponds to 1/4 from the tail to tip. One of "tip", "tail", "cm", "center".
autocolorscale	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
cauto	Determines whether or not the color domain is computed with respect to the input data (here u/v/w norm) or the bounds set in ‘cmin‘ and ‘cmax‘ Defaults to ‘false‘ when ‘cmin‘ and ‘cmax‘ are set by the user. Logical.
cmax	Sets the upper bound of the color domain. Value should have the same units as u/v/w norm and if set, ‘cmin‘ must be set as well. Number.
cmid	Sets the mid-point of the color domain by scaling ‘cmin‘ and/or ‘cmax‘ to be equidistant to this point. Value should have the same units as u/v/w norm. Has no effect when ‘cauto‘ is ‘false‘. Number.
cmin	Sets the lower bound of the color domain. Value should have the same units as u/v/w norm and if set, ‘cmax‘ must be set as well. Number.
coloraxis	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: "coloraxis", "coloraxis2", "coloraxis3", ...
colorbar	Named list or go_cone_colorbar() object.
colorscale	Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘cmin‘ and ‘cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

hoverinfo Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of "x", "y", "z", "u", "v", "w", "norm", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.

hoverlabel Named list or go_cone_hoverlabel() object.

hovertemplate Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variable ‘norm‘ Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example ‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

hovertemplatefallback Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

hovertext Same as ‘text‘. String, or a vector of these.

ids Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

legend Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: "legend", "legend2", "legend3", ...

legendgroup Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

legendgrouptitle Named list or go_cone_legendgrouptitle() object.

legendrank Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on

bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`lighting` Named list or `go_cone_lighting()` object.

`lightposition` Named list or `go_cone_lightposition()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

`name` `go_cone()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_cone_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

`opacity` Sets the opacity of the surface. Please note that in the case of using high 'opacity' values for example a value greater than or equal to 0.5 on two surfaces (and 0.25 with four surfaces), an overlay of multiple transparent surfaces may not perfectly be sorted in depth by the webgl API. This behavior may be improved in the near future and is subject to change. Number in the interval [0, 1].

`reversescale` Reverses the color mapping if true. If true, 'cmin' will correspond to the last color in the array and 'cmax' will correspond to the first color. Logical.

`scene` Sets a reference between this trace's 3D coordinate system and a 3D scene. If "scene" (the default value), the (x,y,z) coordinates refer to 'layout.scene'. If "scene2", the (x,y,z) coordinates refer to 'layout.scene2', and so on. Subplot id: `"scene"`, `"scene2"`, `"scene3"`, ...

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Logical.

`sizemode` Determines whether 'sizeref' is set as a "scaled" (i.e unitless) scalar (normalized by the max u/v/w norm in the vector field) or as "absolute" value (in the same units as the vector field). To display sizes in actual vector length use "raw". One of `"scaled"`, `"absolute"`, `"raw"`.

`sizeref` Adjusts the cone size scaling. The size of the cones is determined by their u/v/w norm multiplied a factor and 'sizeref'. This factor (computed internally) corresponds to the minimum "time" to travel across two successive x/y/z positions at

the average velocity of those two successive positions. All cones in a given trace use the same factor. With ‘sizemode‘ set to "raw", its default value is 1. With ‘sizemode‘ set to "scaled", ‘sizeref‘ is unitless, its default value is 0.5. With ‘sizemode‘ set to "absolute", ‘sizeref‘ has the same units as the u/v/w vector field, its the default value is half the sample’s maximum vector norm. Number in the interval [0, Inf].

`text` `go_cone()`: Sets the text elements associated with the cones. If trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_cone_colorbar_title()`: Sets the title of the color bar. String.

`go_cone_legendgrouptitle()`: Sets the title of the legend group. String.

`u` Sets the x components of the vector field. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`uhoverformat` Sets the hover text formatting rule for ‘u‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`v` Sets the y components of the vector field. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`vhoverformat` Sets the hover text formatting rule for ‘v‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

`visible` Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`w` Sets the z components of the vector field. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`whoverformat` Sets the hover text formatting rule for ‘w‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

x
go_cone(): Sets the x coordinates of the vector field and of the displayed cones. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_cone_colorbar(): Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

go_cone_lightposition(): Numeric vector, representing the X coordinate for each vertex. Number in the interval [-100000, 100000].

xhoverformat
Sets the hover text formatting rule for ‘x‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘xaxis.hoverformat‘. String.

y
go_cone(): Sets the y coordinates of the vector field and of the displayed cones. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_cone_colorbar(): Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

go_cone_lightposition(): Numeric vector, representing the Y coordinate for each vertex. Number in the interval [-100000, 100000].

yhoverformat
Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

z
go_cone(): Sets the z coordinates of the vector field and of the displayed cones. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_cone_lightposition(): Numeric vector, representing the Z coordinate for each vertex. Number in the interval [-100000, 100000].

zhoverformat
Sets the hover text formatting rule for ‘z‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘zaxis.hoverformat‘. String.

skip_invalid — If TRUE, silently drop invalid properties instead of raising an error.

bgcolor — go_cone_colorbar(): Sets the color of padded area. Color: a CSS color string or R color name.

go_cone_hoverlabel(): Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

bordercolor — go_cone_colorbar(): Sets the axis line color. Color: a CSS color string or R color name.

go_cone_hoverlabel(): Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

borderwidth — Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

dtick — Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

exponentformat — Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

labelalias — Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

len — Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

lenmode — Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of "fraction", "pixels".

minexponent — Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

nticks — Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

orientation — Sets the orientation of the colorbar. One of "h", "v".

outlinecolor — Sets the axis line color. Color: a CSS color string or R color name.

outlinewidth — Sets the width (in px) of the axis line. Number in the interval [0, Inf].

separatethousands
: If "true", even 4-digit integers are separated Logical.

showexponent
: If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticklabels
: Determines whether or not the tick labels are drawn. Logical.

showtickprefix
: If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticksuffix
: Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

thickness
: Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode
: Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

tick0
: Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle
: Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

tickcolor
: Sets the tick color. Color: a CSS color string or R color name.

tickfont
: Sets the color bar’s tick label font Named list or `go_cone_colorbar_tickfont()` object.

tickformat
: Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops
: List of named lists or `go_cone_colorbar_tickformatstop()` objects.

ticklabeloverflow
: Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition`
: Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep`
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen`
: Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode`
: Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix`
: Sets a tick label prefix. String.

`ticks`
: Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix`
: Sets a tick label suffix. String.

`ticktext`
: Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals`
: Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth`
: Sets the tick width (in px). Number in the interval [0, Inf].

`title`
: String (the title text), named list or `go_cone_colorbar_title()` object.

`xanchor`
: Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad`
: Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref`
: Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`yanchor`
: Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

ypad
: Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref
: Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

color
: `go_cone_colorbar_tickfont()`, `go_cone_colorbar_title_font()`, `go_cone_legendgrouptitle_f` Color: a CSS color string or R color name.

  `go_cone_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

family
: `go_cone_colorbar_tickfont()`, `go_cone_colorbar_title_font()`, `go_cone_legendgrouptitle_f` HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

  `go_cone_hoverlabel_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

lineposition
: `go_cone_colorbar_tickfont()`, `go_cone_colorbar_title_font()`, `go_cone_legendgrouptitle_f` Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

  `go_cone_hoverlabel_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

shadow
: `go_cone_colorbar_tickfont()`, `go_cone_colorbar_title_font()`, `go_cone_legendgrouptitle_f` Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

  `go_cone_hoverlabel_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size
: `go_cone_colorbar_tickfont()`, `go_cone_colorbar_title_font()`, `go_cone_legendgrouptitle_f` Number in the interval [1, Inf].

  `go_cone_hoverlabel_font()`: Number in the interval [1, Inf], or a vector of these.

style
: `go_cone_colorbar_tickfont()`, `go_cone_colorbar_title_font()`, `go_cone_legendgrouptitle_f` Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

  `go_cone_hoverlabel_font()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`, or a vector of these.

textcase
: go_cone_colorbar_tickfont(), go_cone_colorbar_title_font(), go_cone_legendgrouptitle_f
Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

: go_cone_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant
: go_cone_colorbar_tickfont(), go_cone_colorbar_title_font(), go_cone_legendgrouptitle_f
Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

: go_cone_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight
: go_cone_colorbar_tickfont(), go_cone_colorbar_title_font(), go_cone_legendgrouptitle_f
Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

: go_cone_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange
: range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled
: Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname
: Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value
: string - dtickformat for described zoom level, the same as "tickformat" String.

font
: go_cone_colorbar_title(): Sets this color bar's title font. Named list or go_cone_colorbar_title_font() object.

: go_cone_hoverlabel(): Sets the font used in hover labels. Named list or go_cone_hoverlabel_font() object.

: go_cone_legendgrouptitle(): Sets this legend group's title font. Named list or go_cone_legendgrouptitle_font() object.

side
: Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of "right", "top", "bottom".

align
: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less

than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow — Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.

ambient — Ambient light increases overall color visibility but can wash out the image. Number in the interval [0, 1].

diffuse — Represents the extent that incident rays are reflected in a range of angles. Number in the interval [0, 1].

facenormalsepsilon — Epsilon for face normals calculation avoids math issues arising from degenerate geometry. Number in the interval [0, 1].

fresnel — Represents the reflectance as a dependency of the viewing angle; e.g. paper is reflective when viewing it from the edge of the paper (almost 90 degrees), causing shine. Number in the interval [0, 5].

roughness — Alters specular reflection; the rougher the surface, the wider and less contrasty the shine. Number in the interval [0, 1].

specular — Represents the level that incident rays are reflected in a single direction, causing shine. Number in the interval [0, 2].

vertexnormalsepsilon — Epsilon for vertex normals calculation avoids math issues arising from degenerate geometry. Number in the interval [0, 1].

fig — A `go_figure` object.

data — A data frame in which formula properties (e.g. `x = ~col`, `marker_size = ~col`) are evaluated. Default: the data given to `go_figure()`. Only the evaluated values are added to the trace. A `crosstalk::SharedData` works the same way, and links the trace to its crosstalk group (see `highlight()`).

row — Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

col — Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.

Details

`go_cone()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_cone()` creates an object and adds it to a figure.

Nested objects documented here:

`go_cone_colorbar()` `cone.colorbar`

`go_cone_colorbar_tickfont()` `cone.colorbar.tickfont`: Sets the color bar's tick label font

`go_cone_colorbar_tickformatstop()` `cone.colorbar.tickformatstop`

`go_cone_colorbar_title()` `cone.colorbar.title`

`go_cone_colorbar_title_font()` `cone.colorbar.title.font`: Sets this color bar's title font.

`go_cone_hoverlabel()` `cone.hoverlabel`

`go_cone_hoverlabel_font()` `cone.hoverlabel.font`: Sets the font used in hover labels.

`go_cone_legendgrouptitle()` `cone.legendgrouptitle`

`go_cone_legendgrouptitle_font()` `cone.legendgrouptitle.font`: Sets this legend group's title font.

`go_cone_lighting()` `cone.lighting`

`go_cone_lightposition()` `cone.lightposition`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_cone(x = 1:3, y = c(2, 1, 3))
```

---

go_contour *Contour traces*

Description

The data from which contour lines are computed is set in 'z'. Data in 'z' must be a matrix of numbers. Say that 'z' has N rows and M columns, then by default, these N rows correspond to N y coordinates (set in 'y' or auto-generated) and the M columns correspond to M x coordinates (set in 'x' or auto-generated). By setting 'transpose' to TRUE, the above behavior is flipped.

Usage

```
go_contour(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  autocontour = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  connectgaps = NULL,
  contours = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
```

```
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoverongaps = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  meta = NULL,
  name = NULL,
  ncontours = NULL,
  opacity = NULL,
  reversescale = NULL,
  showlegend = NULL,
  showscale = NULL,
  text = NULL,
  textfont = NULL,
  texttemplate = NULL,
  texttemplatefallback = NULL,
  transpose = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  x = NULL,
  x0 = NULL,
  xaxis = NULL,
  xcalendar = NULL,
  xhoverformat = NULL,
  xperiod = NULL,
  xperiod0 = NULL,
  xperiodalignment = NULL,
  xtype = NULL,
  y = NULL,
  y0 = NULL,
  yaxis = NULL,
  ycalendar = NULL,
  yhoverformat = NULL,
  yperiod = NULL,
  yperiod0 = NULL,
  yperiodalignment = NULL,
  ytype = NULL,
  z = NULL,
```

```
  zauto = NULL,
  zhoverformat = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  zorder = NULL,
  skip_invalid = FALSE
)

go_contour_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
```

```
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_contour_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_contour_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_contour_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)
```

```
    go_contour_colorbar_title_font(
      arg = NULL,
      ...,
      color = NULL,
      family = NULL,
      lineposition = NULL,
      shadow = NULL,
      size = NULL,
      style = NULL,
      textcase = NULL,
      variant = NULL,
      weight = NULL,
      skip_invalid = FALSE
    )

    go_contour_contours(
      arg = NULL,
      ...,
      coloring = NULL,
      end = NULL,
      labelfont = NULL,
      labelformat = NULL,
      operation = NULL,
      showlabels = NULL,
      showlines = NULL,
      size = NULL,
      start = NULL,
      type = NULL,
      value = NULL,
      skip_invalid = FALSE
    )

    go_contour_contours_labelfont(
      arg = NULL,
      ...,
      color = NULL,
      family = NULL,
      lineposition = NULL,
      shadow = NULL,
      size = NULL,
      style = NULL,
      textcase = NULL,
      variant = NULL,
      weight = NULL,
      skip_invalid = FALSE
    )

    go_contour_hoverlabel(
```

```
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_contour_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_contour_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_contour_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

```
go_contour_line(
  arg = NULL,
  ...,
  color = NULL,
  dash = NULL,
  smoothing = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_contour_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

add_contour(fig, ...)

## Default S3 method:
add_contour(
  fig,
  ...,
  data = NULL,
  autocolorscale = NULL,
  autocontour = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  connectgaps = NULL,
  contours = NULL,
  customdata = NULL,
  dx = NULL,
  dy = NULL,
  fillcolor = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hoverongaps = NULL,
  hovertemplate = NULL,
```

```
hovertemplatefallback = NULL,
hovertext = NULL,
ids = NULL,
legend = NULL,
legendgroup = NULL,
legendgrouptitle = NULL,
legendrank = NULL,
legendwidth = NULL,
line = NULL,
meta = NULL,
name = NULL,
ncontours = NULL,
opacity = NULL,
reversescale = NULL,
showlegend = NULL,
showscale = NULL,
text = NULL,
textfont = NULL,
texttemplate = NULL,
texttemplatefallback = NULL,
transpose = NULL,
uid = NULL,
uirevision = NULL,
visible = NULL,
x = NULL,
x0 = NULL,
xaxis = NULL,
xcalendar = NULL,
xhoverformat = NULL,
xperiod = NULL,
xperiod0 = NULL,
xperiodalignment = NULL,
xtype = NULL,
y = NULL,
y0 = NULL,
yaxis = NULL,
ycalendar = NULL,
yhoverformat = NULL,
yperiod = NULL,
yperiod0 = NULL,
yperiodalignment = NULL,
ytype = NULL,
z = NULL,
zauto = NULL,
zhoverformat = NULL,
zmax = NULL,
zmid = NULL,
zmin = NULL,
```

```
  zorder = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
autocolorscale	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
autocontour	Determines whether or not the contour level attributes are picked by an algorithm. If TRUE, the number of contour levels can be set in ‘ncontours‘. If FALSE, set the contour level attributes in ‘contours‘. Logical.
coloraxis	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...
colorbar	Named list or `go_contour_colorbar()` object.
colorscale	Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘zmin‘ and ‘zmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).
connectgaps	Determines whether or not gaps (i.e. {nan} or missing values) in the ‘z‘ data are filled in. It is defaulted to true if ‘z‘ is a one dimensional array otherwise it is defaulted to false. Logical.
contours	Named list or `go_contour_contours()` object.
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.
dx	Sets the x coordinate step. See ‘x0‘ for more info. Number.

dy Sets the y coordinate step. See ‘y0‘ for more info. Number.

fillcolor Sets the fill color if ‘contours.type‘ is "constraint". Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name, or a number mapped to a color by contour.colorscale.

hoverinfo Determines what trace information appears on hover. If ‘none‘ or ‘skip‘ are set, no information is displayed upon hovering. But, if ‘none‘ is set, click and hover events are still fired. Any combination of "x", "y", "z", "text", "name" joined with "+", or one of "all", "none", "skip", or a vector of these.

hoverlabel Named list or go_contour_hoverlabel() object.

hoverongaps Determines whether or not gaps (i.e. {nan} or missing values) in the ‘z‘ data have hover labels associated with them. Logical.

hovertemplate Template string used for rendering the information that appear on hover box. Note that this will override ‘hoverinfo‘. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in ‘hovertemplate‘ are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Anything contained in tag ‘<extra>‘ is displayed in the secondary box, for example ‘<extra>%{fullData.name}</extra>‘. To hide the secondary box completely, use an empty tag ‘<extra></extra>‘. String, or a vector of these.

hovertemplatefallback Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

hovertext Same as ‘text‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

ids Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

legend Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: "legend", "legend2", "legend3", ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
Named list or `go_contour_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line` Named list or `go_contour_line()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

`name` `go_contour()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_contour_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

`ncontours` Sets the maximum number of contour levels. The actual number of contours will be chosen automatically to be less than or equal to the value of 'ncontours'. Has an effect only if 'autocontour' is TRUE or if 'contours.size' is missing. Integer in the interval [1, Inf].

`opacity` Sets the opacity of the trace. Number in the interval [0, 1].

`reversescale` Reverses the color mapping if true. If true, 'zmin' will correspond to the last color in the array and 'zmax' will correspond to the first color. Logical.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Logical.

`text` `go_contour()`: Sets the text elements associated with each z value. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_contour_colorbar_title()`: Sets the title of the color bar. String.

`go_contour_legendgrouptitle()`: Sets the title of the legend group. String.

`textfont` For this trace it only has an effect if ‘contours.coloring‘ is set to "heatmap". Sets the text font. Named list or `go_contour_textfont()` object.

`texttemplate` For this trace it only has an effect if ‘contours.coloring‘ is set to "heatmap". Template string used for rendering the information text that appears on points. Note that this will override ‘textinfo‘. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. All attributes that can be specified per-point (the ones that are ‘arrayOk: true‘) are available. Finally, the template string has access to variables ‘x‘, ‘y‘, ‘z‘ and ‘text‘. String.

`texttemplatefallback`
Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`transpose` Transposes the z data. Logical.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(x|y)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`visible` Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`x` `go_contour()`: Sets the x coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_contour_colorbar()`: Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`x0` Alternate to ‘x‘. Builds a linear space of x coordinates. Use with ‘dx‘ where ‘x0‘ is the starting coordinate and ‘dx‘ the step. Any value.

xaxis — Sets a reference between this trace's x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to 'layout.xaxis'. If "x2", the x coordinates refer to 'layout.xaxis2', and so on. Subplot id: "x", "x2", "x3", ...

xcalendar — Sets the calendar system to use with 'x' date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

xhoverformat — Sets the hover text formatting rule for 'x' using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using 'xaxis.hoverformat'. String.

xperiod — Only relevant when the axis 'type' is "date". Sets the period positioning in milliseconds or "M<n>" on the x axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case 'n' must be a positive integer. Any value.

xperiod0 — Only relevant when the axis 'type' is "date". Sets the base for period positioning in milliseconds or date string on the x0 axis. When 'x0period' is round number of weeks, the 'x0period0' by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

xperiodalignment — Only relevant when the axis 'type' is "date". Sets the alignment of data points on the x axis. One of "start", "middle", "end".

xtype — If "array", the heatmap's x coordinates are given by "x" (the default behavior when 'x' is provided). If "scaled", the heatmap's x coordinates are given by "x0" and "dx" (the default behavior when 'x' is not provided). One of "array", "scaled".

y — go_contour(): Sets the y coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

go_contour_colorbar(): Sets the y position with respect to 'yref' of the color bar (in plot fraction). When 'yref' is "paper", defaults to 0.5 when 'orientation' is "v" and 1.02 when 'orientation' is "h". When 'yref' is "container", defaults to 0.5 when 'orientation' is "v" and 1 when 'orientation' is "h". Must be between 0 and 1 if 'yref' is "container" and between "-2" and 3 if 'yref' is "paper". Number.

y0 — Alternate to 'y'. Builds a linear space of y coordinates. Use with 'dy' where 'y0' is the starting coordinate and 'dy' the step. Any value.

yaxis — Sets a reference between this trace's y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to 'layout.yaxis'. If "y2", the y coordinates refer to 'layout.yaxis2', and so on. Subplot id: "y", "y2", "y3", ...

ycalendar — Sets the calendar system to use with 'y' date data. One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

yhoverformat Sets the hover text formatting rule for ‘y‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46"By default the values are formatted using ‘yaxis.hoverformat‘. String.

yperiod Only relevant when the axis ‘type‘ is "date". Sets the period positioning in milliseconds or "M<n>" on the y axis. Special values in the form of "M<n>" could be used to declare the number of months. In this case ‘n‘ must be a positive integer. Any value.

yperiod0 Only relevant when the axis ‘type‘ is "date". Sets the base for period positioning in milliseconds or date string on the y0 axis. When ‘y0period‘ is round number of weeks, the ‘y0period0‘ by default would be on a Sunday i.e. 2000-01-02, otherwise it would be at 2000-01-01. Any value.

yperiodalignment Only relevant when the axis ‘type‘ is "date". Sets the alignment of data points on the y axis. One of `"start"`, `"middle"`, `"end"`.

ytype If "array", the heatmap’s y coordinates are given by "y" (the default behavior when ‘y‘ is provided) If "scaled", the heatmap’s y coordinates are given by "y0" and "dy" (the default behavior when ‘y‘ is not provided) One of `"array"`, `"scaled"`.

z Sets the z data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

zauto Determines whether or not the color domain is computed with respect to the input data (here in ‘z‘) or the bounds set in ‘zmin‘ and ‘zmax‘ Defaults to ‘false‘ when ‘zmin‘ and ‘zmax‘ are set by the user. Logical.

zhoverformat Sets the hover text formatting rule for ‘z‘ using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format.By default the values are formatted using generic number format. String.

zmax Sets the upper bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmin‘ must be set as well. Number.

zmid Sets the mid-point of the color domain by scaling ‘zmin‘ and/or ‘zmax‘ to be equidistant to this point. Value should have the same units as in ‘z‘. Has no effect when ‘zauto‘ is ‘false‘. Number.

zmin Sets the lower bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmax‘ must be set as well. Number.

zorder Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher ‘zorder‘ appear in front of those with lower ‘zorder‘. Integer.

skip_invalid If `TRUE`, silently drop invalid properties instead of raising an error.

bgcolor `go_contour_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

`go_contour_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`bordercolor` `go_contour_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

`go_contour_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands`
If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticksuffix Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

thickness Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode Determines whether this color bar’s thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use ‘thickness‘ to set the value. One of `"fraction"`, `"pixels"`.

tick0 Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar’s tick label font Named list or `go_contour_colorbar_tickfont()` object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3’s date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or `go_contour_colorbar_tickformatstop()` objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

ticklabelposition Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

ticklabelstep Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows

every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_contour_colorbar_title()` object.

`xanchor` Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`yanchor` Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

`ypad` Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

`yref` Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

color — go_contour_colorbar_tickfont(), go_contour_colorbar_title_font(), go_contour_contours_labelfont(), go_contour_legendgrouptitle_font(), go_contour_textfont(): Color: a CSS color string or R color name.

go_contour_hoverlabel_font(): Color: a CSS color string or R color name, or a vector of these.

go_contour_line(): Sets the color of the contour level. Has no effect if 'contours.coloring' is set to "lines". Color: a CSS color string or R color name.

family — go_contour_colorbar_tickfont(), go_contour_colorbar_title_font(), go_contour_contours_labelfont(), go_contour_legendgrouptitle_font(), go_contour_textfont(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

go_contour_hoverlabel_font(): HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

lineposition — go_contour_colorbar_tickfont(), go_contour_colorbar_title_font(), go_contour_contours_labelfont(), go_contour_legendgrouptitle_font(), go_contour_textfont(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

go_contour_hoverlabel_font(): Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none", or a vector of these.

shadow — go_contour_colorbar_tickfont(), go_contour_colorbar_title_font(), go_contour_contours_labelfont(), go_contour_legendgrouptitle_font(), go_contour_textfont(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

go_contour_hoverlabel_font(): Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size — go_contour_colorbar_tickfont(), go_contour_colorbar_title_font(), go_contour_contours_labelfont(), go_contour_legendgrouptitle_font(), go_contour_textfont(): Number in the interval [1, Inf].

go_contour_contours(): Sets the step between each contour level. Must be positive. Number in the interval [0, Inf].

go_contour_hoverlabel_font(): Number in the interval [1, Inf], or a vector of these.

style — go_contour_colorbar_tickfont(), go_contour_colorbar_title_font(), go_contour_contours_labelfont(), go_contour_legendgrouptitle_font(),

go_contour_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

go_contour_hoverlabel_font(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic", or a vector of these.

textcase go_contour_colorbar_tickfont(), go_contour_colorbar_title_font(), go_contour_contours_labelfont(), go_contour_legendgrouptitle_font(), go_contour_textfont(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

go_contour_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant go_contour_colorbar_tickfont(), go_contour_colorbar_title_font(), go_contour_contours_labelfont(), go_contour_legendgrouptitle_font(), go_contour_textfont(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

go_contour_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight go_contour_colorbar_tickfont(), go_contour_colorbar_title_font(), go_contour_contours_labelfont(), go_contour_legendgrouptitle_font(), go_contour_textfont(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

go_contour_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname

Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value go_contour_colorbar_tickformatstop(): string - dtickformat for described zoom level, the same as "tickformat" String.

go_contour_contours(): Sets the value or values of the constraint boundary. When ‘operation‘ is set to one of the comparison values (‘=,<,>=,>,<=‘) "value" is expected to be a number. When ‘operation‘ is set to one of the interval values (‘[],(),[),(],][,)(,](,)[‘) "value" is expected to be an array of two numbers where the first is the lower bound and the second is the upper bound. Any value.

font
: go_contour_colorbar_title(): Sets this color bar's title font. Named list or go_contour_colorbar_title_font() object.
go_contour_hoverlabel(): Sets the font used in hover labels. Named list or go_contour_hoverlabel_font() object.
go_contour_legendgrouptitle(): Sets this legend group's title font. Named list or go_contour_legendgrouptitle_font() object.

side
: Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of "right", "top", "bottom".

coloring
: Determines the coloring method showing the contour values. If "fill", coloring is done evenly between each contour level If "heatmap", a heatmap gradient coloring is applied between each contour level. If "lines", coloring is done on the contour lines. If "none", no coloring is applied on this trace. One of "fill", "heatmap", "lines", "none".

end
: Sets the end contour level value. Must be more than 'contours.start' Number.

labelfont
: Sets the font used for labeling the contour levels. The default color comes from the lines, if shown. The default family and size come from 'layout.font'. Named list or go_contour_contours_labelfont() object.

labelformat
: Sets the contour label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. String.

operation
: Sets the constraint operation. "=" keeps regions equal to 'value' "<" and "<=" keep regions less than 'value' ">" and ">=" keep regions greater than 'value' "[]", "()", "[)", and "(]" keep regions inside 'value[0]' to 'value[1]' "][", ")(", "](", ")[" keep regions outside 'value[0]' to 'value[1]' Open vs. closed intervals make no difference to constraint display, but all versions are allowed. One of "=", "<", ">=", ">", "<=", "[]", "()", "[)", "(]", "][", ")(", "](", ")[".

showlabels
: Determines whether to label the contour lines with their values. Logical.

showlines
: Determines whether or not the contour lines are drawn. Has an effect only if 'contours.coloring' is set to "fill". Logical.

start
: Sets the starting contour level value. Must be less than 'contours.end' Number.

type
: If 'levels', the data is represented as a contour plot with multiple levels displayed. If 'constraint', the data is represented as constraints with the invalid region shaded as specified by the 'operation' and 'value' parameters. One of "levels", "constraint".

align
: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.

namelength
: Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.

showarrow	Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.
dash	Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.
smoothing	Sets the amount of smoothing for the contour lines, where 0 corresponds to no smoothing. Number in the interval [0, 1.3].
width	Sets the contour line width (in px) Defaults to 0.5 when 'contours.type' is "levels". Defaults to 2 when 'contours.type' is "constraint". Number in the interval [0, Inf].
fig	A `go_figure` object.
data	A data frame in which formula properties (e.g. `x = ~col`, `marker_size = ~col`) are evaluated. Default: the data given to `go_figure()`. Only the evaluated values are added to the trace. A `crosstalk::SharedData` works the same way, and links the trace to its crosstalk group (see `highlight()`).
row	Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.
col	Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.
secondary_y	If `TRUE`, use the secondary y axis of the subplot.

Details

`go_contour()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_contour()` creates an object and adds it to a figure.

Nested objects documented here:

`go_contour_colorbar()` `contour.colorbar`

`go_contour_colorbar_tickfont()` `contour.colorbar.tickfont`: Sets the color bar's tick label font

`go_contour_colorbar_tickformatstop()` `contour.colorbar.tickformatstop`

`go_contour_colorbar_title()` `contour.colorbar.title`

`go_contour_colorbar_title_font()` `contour.colorbar.title.font`: Sets this color bar's title font.

`go_contour_contours()` `contour.contours`

`go_contour_contours_labelfont()` `contour.contours.labelfont`: Sets the font used for labeling the contour levels. The default color comes from the lines, if shown. The default family and size come from 'layout.font'.

`go_contour_hoverlabel()` `contour.hoverlabel`

`go_contour_hoverlabel_font()` `contour.hoverlabel.font`: Sets the font used in hover labels.

`go_contour_legendgrouptitle()` `contour.legendgrouptitle`

`go_contour_legendgrouptitle_font()` `contour.legendgrouptitle.font`: Sets this legend group's title font.

`go_contour_line()` `contour.line`

`go_contour_textfont()` `contour.textfont`: For this trace it only has an effect if 'contours.coloring' is set to "heatmap". Sets the text font.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_contour(x = 1:3, y = c(2, 1, 3))
```

---

go_contourcarpet *Contourcarpet traces*

Description

Plots contours on either the first carpet axis or the carpet axis with a matching 'carpet' attribute. Data 'z' is interpreted as matching that of the corresponding carpet axis.

Usage

```
go_contourcarpet(
  arg = NULL,
  ...,
  a = NULL,
  a0 = NULL,
  atype = NULL,
  autocolorscale = NULL,
  autocontour = NULL,
  b = NULL,
  b0 = NULL,
  btype = NULL,
  carpet = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  contours = NULL,
  customdata = NULL,
  da = NULL,
```

```
  db = NULL,
  fillcolor = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  meta = NULL,
  name = NULL,
  ncontours = NULL,
  opacity = NULL,
  reversescale = NULL,
  showlegend = NULL,
  showscale = NULL,
  text = NULL,
  transpose = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  xaxis = NULL,
  yaxis = NULL,
  z = NULL,
  zauto = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  zorder = NULL,
  skip_invalid = FALSE
)

go_contourcarpet_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
```

```
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_contourcarpet_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
```

```
  weight = NULL,
  skip_invalid = FALSE
)

go_contourcarpet_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_contourcarpet_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_contourcarpet_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_contourcarpet_contours(
  arg = NULL,
  ...,
  coloring = NULL,
  end = NULL,
  labelfont = NULL,
  labelformat = NULL,
  operation = NULL,
  showlabels = NULL,
```

```
  showlines = NULL,
  size = NULL,
  start = NULL,
  type = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_contourcarpet_contours_labelfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_contourcarpet_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_contourcarpet_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_contourcarpet_line(
  arg = NULL,
```

```
  ...,
  color = NULL,
  dash = NULL,
  smoothing = NULL,
  width = NULL,
  skip_invalid = FALSE
)

add_contourcarpet(fig, ...)

## Default S3 method:
add_contourcarpet(
  fig,
  ...,
  data = NULL,
  a = NULL,
  a0 = NULL,
  atype = NULL,
  autocolorscale = NULL,
  autocontour = NULL,
  b = NULL,
  b0 = NULL,
  btype = NULL,
  carpet = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  contours = NULL,
  customdata = NULL,
  da = NULL,
  db = NULL,
  fillcolor = NULL,
  hovertext = NULL,
  ids = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  meta = NULL,
  name = NULL,
  ncontours = NULL,
  opacity = NULL,
  reversescale = NULL,
  showlegend = NULL,
  showscale = NULL,
  text = NULL,
```

```
  transpose = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  xaxis = NULL,
  yaxis = NULL,
  z = NULL,
  zauto = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  zorder = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
a	Sets the x coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
a0	Alternate to ‘x‘. Builds a linear space of x coordinates. Use with ‘dx‘ where ‘x0‘ is the starting coordinate and ‘dx‘ the step. Any value.
atype	If "array", the heatmap’s x coordinates are given by "x" (the default behavior when ‘x‘ is provided). If "scaled", the heatmap’s x coordinates are given by "x0" and "dx" (the default behavior when ‘x‘ is not provided). One of `"array"`, `"scaled"`.
autocolorscale	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
autocontour	Determines whether or not the contour level attributes are picked by an algorithm. If TRUE, the number of contour levels can be set in ‘ncontours‘. If FALSE, set the contour level attributes in ‘contours‘. Logical.
b	Sets the y coordinates. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
b0	Alternate to ‘y‘. Builds a linear space of y coordinates. Use with ‘dy‘ where ‘y0‘ is the starting coordinate and ‘dy‘ the step. Any value.
btype	If "array", the heatmap’s y coordinates are given by "y" (the default behavior when ‘y‘ is provided) If "scaled", the heatmap’s y coordinates are given by

"y0" and "dy" (the default behavior when ‘y‘ is not provided) One of `"array"`, `"scaled"`.

`carpet` The ‘carpet‘ of the carpet axes on which this contour trace lies String.

`coloraxis` Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...

`colorbar` Named list or `go_contourcarpet_colorbar()` object.

`colorscale` Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘zmin‘ and ‘zmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi
Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).

`contours` Named list or `go_contourcarpet_contours()` object.

`customdata` Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`da` Sets the x coordinate step. See ‘x0‘ for more info. Number.

`db` Sets the y coordinate step. See ‘y0‘ for more info. Number.

`fillcolor` Sets the fill color if ‘contours.type‘ is "constraint". Defaults to a half-transparent variant of the line color, marker color, or marker line color, whichever is available. Color: a CSS color string or R color name, or a number mapped to a color by `contourcarpet.colorscale`.

`hovertext` Same as ‘text‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under ‘layout.legend‘, ‘layout.legend2‘, etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
Named list or `go_contourcarpet_legendgrouptitle()` object.

`legendrank` Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on

bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth` Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`line` Named list or `go_contourcarpet_line()` object.

`meta` Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace 'name', graph, axis and colorbar 'title.text', annotation 'text' 'rangeselector', 'updatemenues' and 'sliders' 'label' text all support 'meta'. To access the trace 'meta' values in an attribute in the same trace, simply use '%{meta[i]}' where 'i' is the index or key of the 'meta' item in question. To access trace 'meta' in layout attributes, use '%{data[n[.meta[i]}' where 'i' is the index or key of the 'meta' and 'n' is the trace index. Any value.

`name` `go_contourcarpet()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

`go_contourcarpet_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

`ncontours` Sets the maximum number of contour levels. The actual number of contours will be chosen automatically to be less than or equal to the value of 'ncontours'. Has an effect only if 'autocontour' is TRUE or if 'contours.size' is missing. Integer in the interval [1, Inf].

`opacity` Sets the opacity of the trace. Number in the interval [0, 1].

`reversescale` Reverses the color mapping if true. If true, 'zmin' will correspond to the last color in the array and 'zmax' will correspond to the first color. Logical.

`showlegend` Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale` Determines whether or not a colorbar is displayed for this trace. Logical.

`text` `go_contourcarpet()`: Sets the text elements associated with each z value. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`go_contourcarpet_colorbar_title()`: Sets the title of the color bar. String.

`go_contourcarpet_legendgrouptitle()`: Sets the title of the legend group. String.

`transpose` Transposes the z data. Logical.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: 'constraintrange' in 'parcoords' traces, as well as some 'editable: true' modifications such as

	‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.
`visible`	Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.
`xaxis`	Sets a reference between this trace’s x coordinates and a 2D cartesian x axis. If "x" (the default value), the x coordinates refer to ‘layout.xaxis‘. If "x2", the x coordinates refer to ‘layout.xaxis2‘, and so on. Subplot id: `"x"`, `"x2"`, `"x3"`, ...
`yaxis`	Sets a reference between this trace’s y coordinates and a 2D cartesian y axis. If "y" (the default value), the y coordinates refer to ‘layout.yaxis‘. If "y2", the y coordinates refer to ‘layout.yaxis2‘, and so on. Subplot id: `"y"`, `"y2"`, `"y3"`, ...
`z`	Sets the z data. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
`zauto`	Determines whether or not the color domain is computed with respect to the input data (here in ‘z‘) or the bounds set in ‘zmin‘ and ‘zmax‘ Defaults to ‘false‘ when ‘zmin‘ and ‘zmax‘ are set by the user. Logical.
`zmax`	Sets the upper bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmin‘ must be set as well. Number.
`zmid`	Sets the mid-point of the color domain by scaling ‘zmin‘ and/or ‘zmax‘ to be equidistant to this point. Value should have the same units as in ‘z‘. Has no effect when ‘zauto‘ is ‘false‘. Number.
`zmin`	Sets the lower bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmax‘ must be set as well. Number.
`zorder`	Sets the layer on which this trace is displayed, relative to other SVG traces on the same subplot. SVG traces with higher ‘zorder‘ appear in front of those with lower ‘zorder‘. Integer.
`skip_invalid`	If `TRUE`, silently drop invalid properties instead of raising an error.
`bgcolor`	Sets the color of padded area. Color: a CSS color string or R color name.
`bordercolor`	Sets the axis line color. Color: a CSS color string or R color name.
`borderwidth`	Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].
`dtick`	Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.
`exponentformat`	Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B.

"SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar's length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use 'len' to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_contourcarpet_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops` List of named lists or `go_contourcarpet_colorbar_tickformatstop()` objects.

`ticklabeloverflow` Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition` Determines where tick labels are drawn relative to the ticks. Left and right options are used when ‘orientation‘ is "h", top and bottom when ‘orientation‘ is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

tickprefix — Sets a tick label prefix. String.

ticks — Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of "outside", "inside", "".

ticksuffix — Sets a tick label suffix. String.

ticktext — Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals — Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth — Sets the tick width (in px). Number in the interval [0, Inf].

title — String (the title text), named list or go_contourcarpet_colorbar_title() object.

x — Sets the x position with respect to 'xref' of the color bar (in plot fraction). When 'xref' is "paper", defaults to 1.02 when 'orientation' is "v" and 0.5 when 'orientation' is "h". When 'xref' is "container", defaults to 1 when 'orientation' is "v" and 0.5 when 'orientation' is "h". Must be between 0 and 1 if 'xref' is "container" and between "-2" and 3 if 'xref' is "paper". Number.

xanchor — Sets this color bar's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the color bar. Defaults to "left" when 'orientation' is "v" and "center" when 'orientation' is "h". One of "left", "center", "right".

xpad — Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

xref — Sets the container 'x' refers to. "container" spans the entire 'width' of the plot. "paper" refers to the width of the plotting area only. One of "container", "paper".

y — Sets the y position with respect to 'yref' of the color bar (in plot fraction). When 'yref' is "paper", defaults to 0.5 when 'orientation' is "v" and 1.02 when 'orientation' is "h". When 'yref' is "container", defaults to 0.5 when 'orientation' is "v" and 1 when 'orientation' is "h". Must be between 0 and 1 if 'yref' is "container" and between "-2" and 3 if 'yref' is "paper". Number.

yanchor — Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

color — go_contourcarpet_colorbar_tickfont(), go_contourcarpet_colorbar_title_font(), go_contourcarpet_contours_labelfont(), go_contourcarpet_legendgrouptitle_font(): Color: a CSS color string or R color name.

go_contourcarpet_line(): Sets the color of the contour level. Has no effect if 'contours.coloring' is set to "lines". Color: a CSS color string or R color name.

family — HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition — Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow — Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size — go_contourcarpet_colorbar_tickfont(), go_contourcarpet_colorbar_title_font(), go_contourcarpet_contours_labelfont(), go_contourcarpet_legendgrouptitle_font(): Number in the interval [1, Inf].

go_contourcarpet_contours(): Sets the step between each contour level. Must be positive. Number in the interval [0, Inf].

style — Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase — Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant — Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight — Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

value — go_contourcarpet_colorbar_tickformatstop(): string - dtickformat for described zoom level, the same as "tickformat" String.

go_contourcarpet_contours(): Sets the value or values of the constraint boundary. When 'operation' is set to one of the comparison values ('=,<,>=,>,<=') "value" is expected to be a number. When 'operation' is set to one of the interval values ('[],(),[),(],][,)(,](,)[') "value" is expected to be an array of two numbers where the first is the lower bound and the second is the upper bound. Any value.

font — go_contourcarpet_colorbar_title(): Sets this color bar's title font. Named list or go_contourcarpet_colorbar_title_font() object.
go_contourcarpet_legendgrouptitle(): Sets this legend group's title font. Named list or go_contourcarpet_legendgrouptitle_font() object.

side — Determines the location of color bar's title with respect to the color bar. Defaults to "top" when 'orientation' is "v" and defaults to "right" when 'orientation' is "h". One of "right", "top", "bottom".

coloring — Determines the coloring method showing the contour values. If "fill", coloring is done evenly between each contour level If "lines", coloring is done on the contour lines. If "none", no coloring is applied on this trace. One of "fill", "lines", "none".

end — Sets the end contour level value. Must be more than 'contours.start' Number.

labelfont — Sets the font used for labeling the contour levels. The default color comes from the lines, if shown. The default family and size come from 'layout.font'. Named list or go_contourcarpet_contours_labelfont() object.

labelformat — Sets the contour label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. String.

operation — Sets the constraint operation. "=" keeps regions equal to 'value' "<" and "<=" keep regions less than 'value' ">" and ">=" keep regions greater than 'value' "[]", "()", "[)", and "(]" keep regions inside 'value[0]' to 'value[1]' "][", ")(", "](", ")[" keep regions outside 'value[0]' to 'value[1]' Open vs. closed intervals make no difference to constraint display, but all versions are allowed. One of "=", "<", ">=", ">", "<=", "[]", "()", "[)", "(]", "][", ")(", "](", ")[".

showlabels — Determines whether to label the contour lines with their values. Logical.

showlines — Determines whether or not the contour lines are drawn. Has an effect only if 'contours.coloring' is set to "fill". Logical.

start — Sets the starting contour level value. Must be less than 'contours.end' Number.

type — If 'levels', the data is represented as a contour plot with multiple levels displayed. If 'constraint', the data is represented as constraints with the invalid region shaded as specified by the 'operation' and 'value' parameters. One of "levels", "constraint".

dash — Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$".

smoothing — Sets the amount of smoothing for the contour lines, where 0 corresponds to no smoothing. Number in the interval [0, 1.3].

width — Sets the contour line width (in px) Defaults to 0.5 when 'contours.type' is "levels". Defaults to 2 when 'contours.type' is "constraint". Number in the interval [0, Inf].

fig — A go_figure object.

data — A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated

	values are added to the trace. A `crosstalk::SharedData` works the same way, and links the trace to its crosstalk group (see `highlight()`).
`row`	Subplot grid row (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.
`col`	Subplot grid column (figures created with `make_subplots()` only). May be `"all"` or a vector to address several cells.
`secondary_y`	If `TRUE`, use the secondary y axis of the subplot.

Details

`go_contourcarpet()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`). `add_contourcarpet()` creates an object and adds it to a figure.

Nested objects documented here:

`go_contourcarpet_colorbar()` `contourcarpet.colorbar`

`go_contourcarpet_colorbar_tickfont()` `contourcarpet.colorbar.tickfont`: Sets the color bar's tick label font

`go_contourcarpet_colorbar_tickformatstop()` `contourcarpet.colorbar.tickformatstop`

`go_contourcarpet_colorbar_title()` `contourcarpet.colorbar.title`

`go_contourcarpet_colorbar_title_font()` `contourcarpet.colorbar.title.font`: Sets this color bar's title font.

`go_contourcarpet_contours()` `contourcarpet.contours`

`go_contourcarpet_contours_labelfont()` `contourcarpet.contours.labelfont`: Sets the font used for labeling the contour levels. The default color comes from the lines, if shown. The default family and size come from 'layout.font'.

`go_contourcarpet_legendgrouptitle()` `contourcarpet.legendgrouptitle`

`go_contourcarpet_legendgrouptitle_font()` `contourcarpet.legendgrouptitle.font`: Sets this legend group's title font.

`go_contourcarpet_line()` `contourcarpet.line`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_figure()`, `add_trace()`

Examples

```
go_contourcarpet()
```

go_densitymap *Densitymap traces*

Description

Draws a bivariate kernel density estimation with a Gaussian kernel from ‘lon‘ and ‘lat‘ coordinates and optional ‘z‘ values using a colorscale.

Usage

```
go_densitymap(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  below = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  customdata = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  lat = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lon = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  radius = NULL,
  reversescale = NULL,
  showlegend = NULL,
  showscale = NULL,
  subplot = NULL,
  text = NULL,
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  z = NULL,
  zauto = NULL,
  zmax = NULL,
```

```
  zmid = NULL,
  zmin = NULL,
  skip_invalid = FALSE
)

go_densitymap_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
```

```
  xpad = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_densitymap_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_densitymap_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_densitymap_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_densitymap_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
```

```
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_densitymap_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_densitymap_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_densitymap_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_densitymap_legendgrouptitle_font(
  arg = NULL,
```

```
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

add_densitymap(fig, ...)

## Default S3 method:
add_densitymap(
  fig,
  ...,
  data = NULL,
  autocolorscale = NULL,
  below = NULL,
  coloraxis = NULL,
  colorbar = NULL,
  colorscale = NULL,
  customdata = NULL,
  hoverinfo = NULL,
  hoverlabel = NULL,
  hovertemplate = NULL,
  hovertemplatefallback = NULL,
  hovertext = NULL,
  ids = NULL,
  lat = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  lon = NULL,
  meta = NULL,
  name = NULL,
  opacity = NULL,
  radius = NULL,
  reversescale = NULL,
  showlegend = NULL,
  showscale = NULL,
  subplot = NULL,
  text = NULL,
```

```
  uid = NULL,
  uirevision = NULL,
  visible = NULL,
  z = NULL,
  zauto = NULL,
  zmax = NULL,
  zmid = NULL,
  zmin = NULL,
  row = NULL,
  col = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
autocolorscale	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
below	Determines if the densitymap trace will be inserted before the layer with the specified ID. By default, densitymap traces are placed below the first layer of type symbol If set to ”, the layer will be inserted above every existing layer. String.
coloraxis	Sets a reference to a shared color axis. References to these shared color axes are "coloraxis", "coloraxis2", "coloraxis3", etc. Settings for these shared color axes are set in the layout, under ‘layout.coloraxis‘, ‘layout.coloraxis2‘, etc. Note that multiple color scales can be linked to the same color axis. Subplot id: `"coloraxis"`, `"coloraxis2"`, `"coloraxis3"`, ...
colorbar	Named list or `go_densitymap_colorbar()` object.
colorscale	Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘zmin‘ and ‘zmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).
customdata	Assigns extra data each datum. This may be useful when listening to hover, click and selection events. Note that, "scatter" traces also appends customdata

items in the markers DOM elements Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`hoverinfo` Determines what trace information appears on hover. If 'none' or 'skip' are set, no information is displayed upon hovering. But, if 'none' is set, click and hover events are still fired. Any combination of `"lon"`, `"lat"`, `"z"`, `"text"`, `"name"` joined with `"+"`, or one of `"all"`, `"none"`, `"skip"`, or a vector of these.

`hoverlabel` Named list or `go_densitymap_hoverlabel()` object.

`hovertemplate` Template string used for rendering the information that appear on hover box. Note that this will override 'hoverinfo'. Variables are inserted using %{variable}, for example "y: %{y}" as well as %{xother}, {%_xother}, {%_xother_}, {%xother_}. When showing info for several points, "xother" will be added to those with different x positions from the first point. An underscore before or after "(x|y)other" will add a space on that side, only when this field is shown. Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. The variables available in 'hovertemplate' are the ones emitted as event data described at this link https://plotly.com/javascript/plotlyjs-events/#event-data. Additionally, all attributes that can be specified per-point (the ones that are 'arrayOk: true') are available. Anything contained in tag '<extra>' is displayed in the secondary box, for example '<extra>%{fullData.name}</extra>'. To hide the secondary box completely, use an empty tag '<extra></extra>'. String, or a vector of these.

`hovertemplatefallback` Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

`hovertext` Sets hover text elements associated with each (lon,lat) pair If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (lon,lat) coordinates. To be seen, trace 'hoverinfo' must contain a "text" flag. String, or a vector of these.

`ids` Assigns id labels to each datum. These ids for object constancy of data points during animation. Should be an array of strings, not numbers or any other type. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`lat` Sets the latitude coordinates (in degrees North). Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`legend` Sets the reference to a legend to show this trace in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: `"legend"`, `"legend2"`, `"legend3"`, ...

`legendgroup` Sets the legend group for this trace. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.

`legendgrouptitle`
: Named list or `go_densitymap_legendgrouptitle()` object.

`legendrank`
: Sets the legend rank for this trace. Items and groups with smaller ranks are presented on top/left side while with "reversed" ‘legend.traceorder‘ they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. When having unranked or equal rank items shapes would be displayed after traces i.e. according to their order in data and layout. Number.

`legendwidth`
: Sets the width (in px or fraction) of the legend for this trace. Number in the interval [0, Inf].

`lon`
: Sets the longitude coordinates (in degrees East). Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`meta`
: Assigns extra meta information associated with this trace that can be used in various text attributes. Attributes such as trace ‘name‘, graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘rangeselector‘, ‘updatemenues‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. To access the trace ‘meta‘ values in an attribute in the same trace, simply use ‘%{meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ item in question. To access trace ‘meta‘ in layout attributes, use ‘%{data[n[.meta[i]}‘ where ‘i‘ is the index or key of the ‘meta‘ and ‘n‘ is the trace index. Any value.

`name`
: `go_densitymap()`: Sets the trace name. The trace name appears as the legend item and on hover. String.

  `go_densitymap_colorbar_tickformatstop()`: When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`opacity`
: Sets the opacity of the trace. Number in the interval [0, 1].

`radius`
: Sets the radius of influence of one ‘lon‘ / ‘lat‘ point in pixels. Increasing the value makes the densitymap trace smoother, but less detailed. Number in the interval [1, Inf], or a vector of these.

`reversescale`
: Reverses the color mapping if true. If true, ‘zmin‘ will correspond to the last color in the array and ‘zmax‘ will correspond to the first color. Logical.

`showlegend`
: Determines whether or not an item corresponding to this trace is shown in the legend. Logical.

`showscale`
: Determines whether or not a colorbar is displayed for this trace. Logical.

`subplot`
: Sets a reference between this trace's data coordinates and a map subplot. If "map" (the default value), the data refer to ‘layout.map‘. If "map2", the data refer to ‘layout.map2‘, and so on. Subplot id: `"map"`, `"map2"`, `"map3"`, ...

`text`
: `go_densitymap()`: Sets text elements associated with each (lon,lat) pair If a single string, the same string appears over all the data points. If an array of string, the items are mapped in order to the this trace's (lon,lat) coordinates. If

trace ‘hoverinfo‘ contains a "text" flag and "hovertext" is not set, these elements will be seen in the hover labels. String, or a vector of these.

`go_densitymap_colorbar_title()`: Sets the title of the color bar. String.

`go_densitymap_legendgrouptitle()`: Sets the title of the legend group. String.

`uid` Assign an id to this trace, Use this to provide object constancy between traces during animations and transitions. String.

`uirevision` Controls persistence of some user-driven changes to the trace: ‘constraintrange‘ in ‘parcoords‘ traces, as well as some ‘editable: true‘ modifications such as ‘name‘ and ‘colorbar.title‘. Defaults to ‘layout.uirevision‘. Note that other user-driven trace attribute changes are controlled by ‘layout‘ attributes: ‘trace.visible‘ is controlled by ‘layout.legend.uirevision‘, ‘selectedpoints‘ is controlled by ‘layout.selectionrevision‘, and ‘colorbar.(xly)‘ (accessible with ‘config: {editable: true}‘) is controlled by ‘layout.editrevision‘. Trace changes are tracked by ‘uid‘, which only falls back on trace index if no ‘uid‘ is provided. So if your app can add/remove traces before the end of the ‘data‘ array, such that the same trace has a different index, you can still preserve user-driven changes if you give each trace a ‘uid‘ that stays with it as it moves. Any value.

`visible` Determines whether or not this trace is visible. If "legendonly", the trace is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of `"legendonly"`, `TRUE`, `FALSE`.

`z` Sets the points’ weight. For example, a value of 10 would be equivalent to having 10 points of weight 1 in the same spot Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`zauto` Determines whether or not the color domain is computed with respect to the input data (here in ‘z‘) or the bounds set in ‘zmin‘ and ‘zmax‘ Defaults to ‘false‘ when ‘zmin‘ and ‘zmax‘ are set by the user. Logical.

`zmax` Sets the upper bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmin‘ must be set as well. Number.

`zmid` Sets the mid-point of the color domain by scaling ‘zmin‘ and/or ‘zmax‘ to be equidistant to this point. Value should have the same units as in ‘z‘. Has no effect when ‘zauto‘ is ‘false‘. Number.

`zmin` Sets the lower bound of the color domain. Value should have the same units as in ‘z‘ and if set, ‘zmax‘ must be set as well. Number.

`skip_invalid` If `TRUE`, silently drop invalid properties instead of raising an error.

`bgcolor` `go_densitymap_colorbar()`: Sets the color of padded area. Color: a CSS color string or R color name.

`go_densitymap_hoverlabel()`: Sets the background color of the hover labels for this trace Color: a CSS color string or R color name, or a vector of these.

`bordercolor` `go_densitymap_colorbar()`: Sets the axis line color. Color: a CSS color string or R color name.

`go_densitymap_hoverlabel()`: Sets the border color of the hover labels for this trace. Color: a CSS color string or R color name, or a vector of these.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

dtick Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

exponentformat Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

labelalias Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

len Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

lenmode Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of "fraction", "pixels".

minexponent Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

nticks Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

orientation Sets the orientation of the colorbar. One of "h", "v".

outlinecolor Sets the axis line color. Color: a CSS color string or R color name.

outlinewidth Sets the width (in px) of the axis line. Number in the interval [0, Inf].

separatethousands If "true", even 4-digit integers are separated Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".

showticksuffix Same as ‘showtickprefix‘ but for tick suffixes. One of "all", "first", "last", "none".

`thickness` Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

`thicknessmode` Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont` Sets the color bar's tick label font Named list or `go_densitymap_colorbar_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops` List of named lists or `go_densitymap_colorbar_tickformatstop()` objects.

`ticklabeloverflow` Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

`ticklabelposition` Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelstep` Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_densitymap_colorbar_title()` object.

`x` Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`xanchor` Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`y` Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

`yanchor` Sets this color bar’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when ‘orientation‘ is "v" and "bottom" when ‘orientation‘ is "h". One of `"top"`, `"middle"`, `"bottom"`.

`ypad` Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref
: Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

color
: `go_densitymap_colorbar_tickfont()`, `go_densitymap_colorbar_title_font()`, `go_densitymap_legendgrouptitle_font()`: Color: a CSS color string or R color name.

  `go_densitymap_hoverlabel_font()`: Color: a CSS color string or R color name, or a vector of these.

family
: `go_densitymap_colorbar_tickfont()`, `go_densitymap_colorbar_title_font()`, `go_densitymap_legendgrouptitle_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

  `go_densitymap_hoverlabel_font()`: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String, or a vector of these.

lineposition
: `go_densitymap_colorbar_tickfont()`, `go_densitymap_colorbar_title_font()`, `go_densitymap_legendgrouptitle_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

  `go_densitymap_hoverlabel_font()`: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`, or a vector of these.

shadow
: `go_densitymap_colorbar_tickfont()`, `go_densitymap_colorbar_title_font()`, `go_densitymap_legendgrouptitle_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

  `go_densitymap_hoverlabel_font()`: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String, or a vector of these.

size
: `go_densitymap_colorbar_tickfont()`, `go_densitymap_colorbar_title_font()`, `go_densitymap_legendgrouptitle_font()`: Number in the interval [1, Inf].

  `go_densitymap_hoverlabel_font()`: Number in the interval [1, Inf], or a vector of these.

style
: `go_densitymap_colorbar_tickfont()`, `go_densitymap_colorbar_title_font()`, `go_densitymap_legendgrouptitle_font()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

  `go_densitymap_hoverlabel_font()`: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`, or a vector of these.

textcase
: go_densitymap_colorbar_tickfont(), go_densitymap_colorbar_title_font(), go_densitymap_legendgrouptitle_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

: go_densitymap_hoverlabel_font(): Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower", or a vector of these.

variant
: go_densitymap_colorbar_tickfont(), go_densitymap_colorbar_title_font(), go_densitymap_legendgrouptitle_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

: go_densitymap_hoverlabel_font(): Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase", or a vector of these.

weight
: go_densitymap_colorbar_tickfont(), go_densitymap_colorbar_title_font(), go_densitymap_legendgrouptitle_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

: go_densitymap_hoverlabel_font(): Sets the weight (or boldness) of the font. Integer in the interval [1, 1000], or a vector of these.

dtickrange
: range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled
: Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

templateitemname
: Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

value
: string - dtickformat for described zoom level, the same as "tickformat" String.

font
: go_densitymap_colorbar_title(): Sets this color bar's title font. Named list or go_densitymap_colorbar_title_font() object.

: go_densitymap_hoverlabel(): Sets the font used in hover labels. Named list or go_densitymap_hoverlabel_font() object.

: go_densitymap_legendgrouptitle(): Sets this legend group's title font. Named list or go_densitymap_legendgrouptitle_font() object.

side
: Determines the location of color bar's title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of "right", "top", "bottom".

align
: Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of "left", "right", "auto", or a vector of these.

namelength	Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf], or a vector of these.
showarrow	Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.
fig	A go_figure object.
data	A data frame in which formula properties (e.g. x = ~col, marker_size = ~col) are evaluated. Default: the data given to go_figure(). Only the evaluated values are added to the trace. A crosstalk::SharedData works the same way, and links the trace to its crosstalk group (see highlight()).
row	Subplot grid row (figures created with make_subplots() only). May be "all" or a vector to address several cells.
col	Subplot grid column (figures created with make_subplots() only). May be "all" or a vector to address several cells.

Details

go_densitymap() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()). add_densitymap() creates an object and adds it to a figure.

Nested objects documented here:

go_densitymap_colorbar() densitymap.colorbar

go_densitymap_colorbar_tickfont() densitymap.colorbar.tickfont: Sets the color bar's tick label font

go_densitymap_colorbar_tickformatstop() densitymap.colorbar.tickformatstop

go_densitymap_colorbar_title() densitymap.colorbar.title

go_densitymap_colorbar_title_font() densitymap.colorbar.title.font: Sets this color bar's title font.

go_densitymap_hoverlabel() densitymap.hoverlabel

go_densitymap_hoverlabel_font() densitymap.hoverlabel.font: Sets the font used in hover labels.

go_densitymap_legendgrouptitle() densitymap.legendgrouptitle

go_densitymap_legendgrouptitle_font() densitymap.legendgrouptitle.font: Sets this legend group's title font.

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), add_trace()

Examples

```
go_densitymap()
```

---

go_figure *Create a figure*

Description

A figure holds a list of traces (data) and a layout. Everything is validated against the plotly.js schema. Printing a figure renders it as an htmlwidget.

Usage

```
go_figure(data = NULL, layout = NULL, frames = NULL, skip_invalid = FALSE)
```

Arguments

data	A trace (e.g. from go_scatter()), a list of traces, or a figure (or a list with data and layout elements) to copy. Traces may be graph objects or named lists with a type element. Or a data frame: the figure starts without traces, and formula properties of later add_<trace>() calls (e.g. add_scatter(x = ~col)) are evaluated in it. The data frame is kept with the figure in R (attribute "plotly_data") but is not sent to the browser; only the values traces use are. A crosstalk::SharedData works the same way, and the traces whose points are its rows are linked to its crosstalk group (see highlight()).
layout	A go_layout() object or a named list of layout properties.
frames	Animation frames: a list of named lists with name, data (a list of traces), and optionally layout, group, traces and baseframe elements (see https://plotly.com/javascript/animations/).
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.

Value

A go_figure object.

See Also

make_subplots(), add_trace(), update_layout(), update_traces()

Examples

```
fig <- go_figure(
  data = go_scatter(x = 1:3, y = c(2, 1, 3)),
  layout = go_layout(title = "A figure")
)
fig <- fig |>
  add_scatter(x = 1:3, y = c(3, 2, 1), mode = "markers") |>
  update_layout(xaxis_title = "x")
if (interactive()) print(fig)

# pipe in a data frame and refer to its columns with formulas
px_data_iris() |>
  go_figure() |>
  add_scatter(x = ~sepal_length, y = ~sepal_width, mode = "markers", name = "sepal") |>
  add_scatter(x = ~petal_length, y = ~petal_width, mode = "markers", name = "petal")
```

---

go_layout *Figure layout*

Description

Layout properties of a figure: axes, title, legend, margins, fonts and other settings that are not specific to a trace.

Usage

```
go_layout(
  arg = NULL,
  ...,
  activeselection = NULL,
  activeshape = NULL,
  annotations = NULL,
  autosize = NULL,
  autotypenumbers = NULL,
  barcornerradius = NULL,
  bargap = NULL,
  bargroupgap = NULL,
  barmode = NULL,
  barnorm = NULL,
  boxgap = NULL,
  boxgroupgap = NULL,
  boxmode = NULL,
  calendar = NULL,
  clickanywhere = NULL,
  clickmode = NULL,
  coloraxis = NULL,
  colorscale = NULL,
  colorway = NULL,
```

```
    computed = NULL,
    datarevision = NULL,
    dragmode = NULL,
    editrevision = NULL,
    extendfunnelareacolors = NULL,
    extendiciclecolors = NULL,
    extendpiecolors = NULL,
    extendsunburstcolors = NULL,
    extendtreemapcolors = NULL,
    font = NULL,
    funnelareacolorway = NULL,
    funnelgap = NULL,
    funnelgroupgap = NULL,
    funnelmode = NULL,
    geo = NULL,
    grid = NULL,
    height = NULL,
    hiddenlabels = NULL,
    hoveranywhere = NULL,
    hoverdistance = NULL,
    hoverlabel = NULL,
    hovermode = NULL,
    hoversort = NULL,
    hoversubplots = NULL,
    iciclecolorway = NULL,
    images = NULL,
    legend = NULL,
    map = NULL,
    margin = NULL,
    meta = NULL,
    minreducedheight = NULL,
    minreducedwidth = NULL,
    modebar = NULL,
    newselection = NULL,
    newshape = NULL,
    paper_bgcolor = NULL,
    piecolorway = NULL,
    plot_bgcolor = NULL,
    polar = NULL,
    scattergap = NULL,
    scattermode = NULL,
    scene = NULL,
    selectdirection = NULL,
    selectionrevision = NULL,
    selections = NULL,
    separators = NULL,
    shapes = NULL,
    showlegend = NULL,
```

```
  sliders = NULL,
  smith = NULL,
  spikedistance = NULL,
  sunburstcolorway = NULL,
  template = NULL,
  ternary = NULL,
  title = NULL,
  transition = NULL,
  treemapcolorway = NULL,
  uirevision = NULL,
  uniformtext = NULL,
  updatemenus = NULL,
  violingap = NULL,
  violingroupgap = NULL,
  violinmode = NULL,
  waterfallgap = NULL,
  waterfallgroupgap = NULL,
  waterfallmode = NULL,
  width = NULL,
  xaxis = NULL,
  yaxis = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
activeselection	Named list or `go_layout_activeselection()` object.
activeshape	Named list or `go_layout_activeshape()` object.
annotations	List of named lists or `go_layout_annotation()` objects.
autosize	Determines whether or not a layout width or height that has been left undefined by the user is initialized on each relayout. Note that, regardless of this attribute, an undefined layout width or height is always initialized on the first call to plot. Logical.
autotypenumbers	Using "strict" a numeric string in trace data is not converted to a number. Using "convert types" a numeric string in trace data may be treated as a number during automatic axis ‘type‘ detection. This is the default value; however it could be overridden for individual axes. One of `"convert types"`, `"strict"`.
barcornerradius	Sets the rounding of bar corners. May be an integer number of pixels, or a percentage of bar width (as a string ending in %). Any value.

bargap — Sets the gap (in plot fraction) between bars of adjacent location coordinates. Number in the interval [0, 1].

bargroupgap — Sets the gap (in plot fraction) between bars of the same location coordinate. Number in the interval [0, 1].

barmode — Determines how bars at the same location coordinate are displayed on the graph. With "stack", the bars are stacked on top of one another With "relative", the bars are stacked on top of one another, with negative values below the axis, positive values above With "group", the bars are plotted next to one another centered around the shared location. With "overlay", the bars are plotted over one another, you might need to reduce "opacity" to see multiple bars. One of `"stack"`, `"group"`, `"overlay"`, `"relative"`.

barnorm — Sets the normalization for bar traces on the graph. With "fraction", the value of each bar is divided by the sum of all values at that location coordinate. "percent" is the same but multiplied by 100 to show percentages. One of `""`, `"fraction"`, `"percent"`.

boxgap — Sets the gap (in plot fraction) between boxes of adjacent location coordinates. Has no effect on traces that have "width" set. Number in the interval [0, 1].

boxgroupgap — Sets the gap (in plot fraction) between boxes of the same location coordinate. Has no effect on traces that have "width" set. Number in the interval [0, 1].

boxmode — Determines how boxes at the same location coordinate are displayed on the graph. If "group", the boxes are plotted next to one another centered around the shared location. If "overlay", the boxes are plotted over one another, you might need to set "opacity" to see them multiple boxes. Has no effect on traces that have "width" set. One of `"group"`, `"overlay"`.

calendar — Sets the default calendar system to use for interpreting and displaying dates throughout the plot. One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

clickanywhere — If true, ‘plotly_click‘ events will fire for any click position within the plot area, not just over traces. When clicking where there is no trace data, the event will have an empty ‘points‘ array but will include ‘xvals‘ and ‘yvals‘ with click coordinates in data space, and ‘xPixel‘ and ‘yPixel‘ with click coordinates in pixels, relative to the top-left corner of the graph div. Logical.

clickmode — Determines the mode of single click interactions. "event" is the default value and emits the ‘plotly_click‘ event. In addition this mode emits the ‘plotly_selected‘ event in drag modes "lasso" and "select", but with no event data attached (kept for compatibility reasons). The "select" flag enables selecting single data points via click. This mode also supports persistent selections, meaning that pressing Shift while clicking, adds to / subtracts from an existing selection. "select" with ‘hovermode‘: "x" can be confusing, consider explicitly setting ‘hovermode‘: "closest" when using this feature. Selection events are sent accordingly as long as "event" flag is set as well. When the "event" flag is missing, ‘plotly_click‘ and ‘plotly_selected‘ events are not fired. Any combination of `"event"`, `"select"` joined with `"+"`, or one of `"none"`.

coloraxis — Named list or `go_layout_coloraxis()` object.

`colorscale` Named list or `go_layout_colorscale()` object.

`colorway` Sets the default trace colors. Vector of colors.

`computed` Placeholder for exporting automargin-impacting values namely ‘margin.t‘, ‘margin.b‘, ‘margin.l‘ and ‘margin.r‘ in "full-json" mode. Any value.

`datarevision` If provided, a changed value tells ‘Plotly.react‘ that one or more data arrays has changed. This way you can modify arrays in-place rather than making a complete new copy for an incremental change. If NOT provided, ‘Plotly.react‘ assumes that data arrays are being treated as immutable, thus any data array with a different identity from its predecessor contains new data. Any value.

`dragmode` Determines the mode of drag interactions. "select" and "lasso" apply only to scatter traces with markers or text. "orbit" and "turntable" apply only to 3D scenes. One of `"zoom"`, `"pan"`, `"select"`, `"lasso"`, `"drawclosedpath"`, `"drawopenpath"`, `"drawline"`, `"drawrect"`, `"drawcircle"`, `"orbit"`, `"turntable"`, `FALSE`.

`editrevision` Controls persistence of user-driven changes in ‘editable: true‘ configuration, other than trace names and axis titles. Defaults to ‘layout.uirevision‘. Any value.

`extendfunnelareacolors`
If ‘true‘, the funnelarea slice colors (whether given by ‘funnelareacolorway‘ or inherited from ‘colorway‘) will be extended to three times its original length by first repeating every color 20% lighter then each color 20% darker. This is intended to reduce the likelihood of reusing the same color when you have many slices, but you can set ‘false‘ to disable. Colors provided in the trace, using ‘marker.colors‘, are never extended. Logical.

`extendiciclecolors`
If ‘true‘, the icicle slice colors (whether given by ‘iciclecolorway‘ or inherited from ‘colorway‘) will be extended to three times its original length by first repeating every color 20% lighter then each color 20% darker. This is intended to reduce the likelihood of reusing the same color when you have many slices, but you can set ‘false‘ to disable. Colors provided in the trace, using ‘marker.colors‘, are never extended. Logical.

`extendpiecolors`
If ‘true‘, the pie slice colors (whether given by ‘piecolorway‘ or inherited from ‘colorway‘) will be extended to three times its original length by first repeating every color 20% lighter then each color 20% darker. This is intended to reduce the likelihood of reusing the same color when you have many slices, but you can set ‘false‘ to disable. Colors provided in the trace, using ‘marker.colors‘, are never extended. Logical.

`extendsunburstcolors`
If ‘true‘, the sunburst slice colors (whether given by ‘sunburstcolorway‘ or inherited from ‘colorway‘) will be extended to three times its original length by first repeating every color 20% lighter then each color 20% darker. This is intended to reduce the likelihood of reusing the same color when you have many slices, but you can set ‘false‘ to disable. Colors provided in the trace, using ‘marker.colors‘, are never extended. Logical.

`extendtreemapcolors`
If ‘true‘, the treemap slice colors (whether given by ‘treemapcolorway‘ or inherited from ‘colorway‘) will be extended to three times its original length by

first repeating every color 20% lighter then each color 20% darker. This is intended to reduce the likelihood of reusing the same color when you have many slices, but you can set 'false' to disable. Colors provided in the trace, using 'marker.colors', are never extended. Logical.

`font` Sets the global font. Note that fonts used in traces and other layout components inherit from the global font. Named list or `go_layout_font()` object.

`funnelareacolorway` Sets the default funnelarea slice colors. Defaults to the main 'colorway' used for trace colors. If you specify a new list here it can still be extended with lighter and darker colors, see 'extendfunnelareacolors'. Vector of colors.

`funnelgap` Sets the gap (in plot fraction) between bars of adjacent location coordinates. Number in the interval [0, 1].

`funnelgroupgap` Sets the gap (in plot fraction) between bars of the same location coordinate. Number in the interval [0, 1].

`funnelmode` Determines how bars at the same location coordinate are displayed on the graph. With "stack", the bars are stacked on top of one another With "group", the bars are plotted next to one another centered around the shared location. With "overlay", the bars are plotted over one another, you might need to reduce "opacity" to see multiple bars. One of `"stack"`, `"group"`, `"overlay"`.

`geo` Named list or `go_layout_geo()` object.

`grid` Named list or `go_layout_grid()` object.

`height` Sets the plot's height (in px). Number in the interval [10, Inf].

`hiddenlabels` hiddenlabels is the funnelarea & pie chart analog of visible:'legendonly' but it can contain many labels, and can simultaneously hide slices from several pies/funnelarea charts Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`hoveranywhere` If true, 'plotly_hover' events will fire for any cursor position within the plot area, not just over traces. When the cursor is not over a trace, the event will have an empty 'points' array but will include 'xvals' and 'yvals' with cursor coordinates in data space, and 'xPixel' and 'yPixel' with cursor coordinates in pixels, relative to the top-left corner of the graph div. A 'plotly_unhover' event fires when the cursor leaves the plot area. Logical.

`hoverdistance` Sets the default distance (in pixels) to look for data to add hover labels (-1 means no cutoff, 0 means no looking for data). This is only a real distance for hovering on point-like objects, like scatter points. For area-like objects (bars, scatter fills, etc) hovering is on inside the area and off outside, but these objects will not supersede hover on point-like objects in case of conflict. Integer in the interval [-1, Inf].

`hoverlabel` Named list or `go_layout_hoverlabel()` object.

`hovermode` Determines the mode of hover interactions. If "closest", a single hoverlabel will appear for the "closest" point within the 'hoverdistance'. If "x" (or "y"), multiple hoverlabels will appear for multiple points at the "closest" x- (or y-) coordinate within the 'hoverdistance', with the caveat that no more than one hoverlabel will appear per trace. If "x unified" (or "y unified"), a single hoverlabel will

appear multiple points at the closest x- (or y-) coordinate within the ‘hoverdistance‘ with the caveat that no more than one hoverlabel will appear per trace. In this mode, spikelines are enabled by default perpendicular to the specified axis. If false, hover interactions are disabled. One of "x", "y", "closest", "x unified", "y unified", FALSE.

hoversort — Determines the order of items shown in unified hover labels. If "trace", items are sorted by trace index. If "value descending", items are sorted by value from largest to smallest. If "value ascending", items are sorted by value from smallest to largest. Only applies when ‘hovermode‘ is "x unified" or "y unified". One of "trace", "value descending", "value ascending".

hoversubplots — Determines expansion of hover effects to other subplots If "single" just the axis pair of the primary point is included without overlaying subplots. If "overlaying" all subplots using the main axis and occupying the same space are included. If "axis", also include stacked subplots using the same axis when ‘hovermode‘ is set to "x", "x unified", "y" or "y unified". One of "single", "overlaying", "axis".

iciclecolorway — Sets the default icicle slice colors. Defaults to the main ‘colorway‘ used for trace colors. If you specify a new list here it can still be extended with lighter and darker colors, see ‘extendiciclecolors‘. Vector of colors.

images — List of named lists or go_layout_image() objects.

legend — Named list or go_layout_legend() object.

map — Named list or go_layout_map() object.

margin — Named list or go_layout_margin() object.

meta — Assigns extra meta information that can be used in various ‘text‘ attributes. Attributes such as the graph, axis and colorbar ‘title.text‘, annotation ‘text‘ ‘trace.name‘ in legend items, ‘rangeselector‘, ‘updatemenus‘ and ‘sliders‘ ‘label‘ text all support ‘meta‘. One can access ‘meta‘ fields using template strings: ‘%{meta[i]}‘ where ‘i‘ is the index of the ‘meta‘ item in question. ‘meta‘ can also be an object for example ‘{key: value}‘ which can be accessed %{meta[key]}. Any value.

minreducedheight — Minimum height of the plot with margin.automargin applied (in px) Number in the interval [2, Inf].

minreducedwidth — Minimum width of the plot with margin.automargin applied (in px) Number in the interval [2, Inf].

modebar — Named list or go_layout_modebar() object.

newselection — Named list or go_layout_newselection() object.

newshape — Named list or go_layout_newshape() object.

paper_bgcolor — Sets the background color of the paper where the graph is drawn. Color: a CSS color string or R color name.

piecolorway — Sets the default pie slice colors. Defaults to the main ‘colorway‘ used for trace colors. If you specify a new list here it can still be extended with lighter and darker colors, see ‘extendpiecolors‘. Vector of colors.

plot_bgcolor Sets the background color of the plotting area in-between x and y axes. Color: a CSS color string or R color name.

polar Named list or go_layout_polar() object.

scattergap Sets the gap (in plot fraction) between scatter points of adjacent location coordinates. Defaults to ‘bargap‘. Number in the interval [0, 1].

scattermode Determines how scatter points at the same location coordinate are displayed on the graph. With "group", the scatter points are plotted next to one another centered around the shared location. With "overlay", the scatter points are plotted over one another, you might need to reduce "opacity" to see multiple scatter points. One of "group", "overlay".

scene Named list or go_layout_scene() object.

selectdirection When ‘dragmode‘ is set to "select", this limits the selection of the drag to horizontal, vertical or diagonal. "h" only allows horizontal selection, "v" only vertical, "d" only diagonal and "any" sets no limit. One of "h", "v", "d", "any".

selectionrevision Controls persistence of user-driven changes in selected points from all traces. Any value.

selections List of named lists or go_layout_selection() objects.

separators Sets the decimal and thousand separators. For example, ". " puts a ’.’ before decimals and a space between thousands. In English locales, dflt is "." but other locales may alter this default. String.

shapes List of named lists or go_layout_shape() objects.

showlegend Determines whether or not a legend is drawn. Default is ‘true‘ if there is a trace to show and any of these: a) Two or more traces would by default be shown in the legend. b) One pie trace is shown in the legend. c) One trace is explicitly given with ‘showlegend: true‘. Logical.

sliders List of named lists or go_layout_slider() objects.

smith Named list or go_layout_smith() object.

spikedistance Sets the default distance (in pixels) to look for data to draw spikelines to (-1 means no cutoff, 0 means no looking for data). As with hoverdistance, distance does not apply to area-like objects. In addition, some objects can be hovered on but will not generate spikelines, such as scatter fills. Integer in the interval [-1, Inf].

sunburstcolorway Sets the default sunburst slice colors. Defaults to the main ‘colorway‘ used for trace colors. If you specify a new list here it can still be extended with lighter and darker colors, see ‘extendsunburstcolors‘. Vector of colors.

template Default attributes to be applied to the plot. Templates can be created from existing plots using ‘Plotly.makeTemplate‘, or created manually. They should be objects with format: ‘{layout: layoutTemplate, data: {[type]: [traceTemplate, ...]}, ...}‘ ‘layoutTemplate‘ and ‘traceTemplate‘ are objects matching the attribute structure of ‘layout‘ and a data trace. Trace templates are applied cyclically to traces of each type. Container arrays (eg ‘annotations‘) have special

handling: An object ending in ‘defaults‘ (eg ‘annotationdefaults‘) is applied to each array item. But if an item has a ‘templateitemname‘ key we look in the template array for an item with matching ‘name‘ and apply that instead. If no matching ‘name‘ is found we mark the item invisible. Any named template item not referenced is appended to the end of the array, so you can use this for a watermark annotation or a logo image, for example. To omit one of these items on the plot, make an item with matching ‘templateitemname‘ and ‘visible: false‘. Any value.

`ternary` Named list or `go_layout_ternary()` object.

`title` String (the title text), named list or `go_layout_title()` object.

`transition` Sets transition options used during Plotly.react updates. Named list or `go_layout_transition()` object.

`treemapcolorway`
Sets the default treemap slice colors. Defaults to the main ‘colorway‘ used for trace colors. If you specify a new list here it can still be extended with lighter and darker colors, see ‘extendtreemapcolors‘. Vector of colors.

`uirevision` Used to allow user interactions with the plot to persist after ‘Plotly.react‘ calls that are unaware of these interactions. If ‘uirevision‘ is omitted, or if it is given and it changed from the previous ‘Plotly.react‘ call, the exact new figure is used. If ‘uirevision‘ is truthy and did NOT change, any attribute that has been affected by user interactions and did not receive a different value in the new figure will keep the interaction value. ‘layout.uirevision‘ attribute serves as the default for ‘uirevision‘ attributes in various sub-containers. For finer control you can set these sub-attributes directly. For example, if your app separately controls the data on the x and y axes you might set ‘xaxis.uirevision="time"‘ and ‘yaxis.uirevision="cost"‘. Then if only the y data is changed, you can update ‘yaxis.uirevision="quantity"‘ and the y axis range will reset but the x axis range will retain any user-driven zoom. Any value.

`uniformtext` Named list or `go_layout_uniformtext()` object.

`updatemenus` List of named lists or `go_layout_updatemenu()` objects.

`violingap` Sets the gap (in plot fraction) between violins of adjacent location coordinates. Has no effect on traces that have "width" set. Number in the interval [0, 1].

`violingroupgap` Sets the gap (in plot fraction) between violins of the same location coordinate. Has no effect on traces that have "width" set. Number in the interval [0, 1].

`violinmode` Determines how violins at the same location coordinate are displayed on the graph. If "group", the violins are plotted next to one another centered around the shared location. If "overlay", the violins are plotted over one another, you might need to set "opacity" to see them multiple violins. Has no effect on traces that have "width" set. One of `"group"`, `"overlay"`.

`waterfallgap` Sets the gap (in plot fraction) between bars of adjacent location coordinates. Number in the interval [0, 1].

`waterfallgroupgap`
Sets the gap (in plot fraction) between bars of the same location coordinate. Number in the interval [0, 1].

waterfallmode	Determines how bars at the same location coordinate are displayed on the graph. With "group", the bars are plotted next to one another centered around the shared location. With "overlay", the bars are plotted over one another, you might need to reduce "opacity" to see multiple bars. One of "group", "overlay".
width	Sets the plot's width (in px). Number in the interval [10, Inf].
xaxis	Named list or go_layout_xaxis() object.
yaxis	Named list or go_layout_yaxis() object.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.

Details

go_layout() create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_figure(), update_layout()

Examples

```
go_layout(title = "My plot", xaxis = list(title = "x", range = c(0, 10)))
```

---

go_layout_activeselection

Layout activeselection objects

Description

Properties of layout.activeselection.

Usage

```
go_layout_activeselection(
  arg = NULL,
  ...,
  fillcolor = NULL,
  opacity = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
fillcolor	Sets the color filling the active selection' interior. Color: a CSS color string or R color name.
opacity	Sets the opacity of the active selection. Number in the interval [0, 1].
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.

Details

go_layout_activeselection() create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

---

go_layout_activeshape *Layout activeshape objects*

Description

Properties of layout.activeshape.

Usage

```
go_layout_activeshape(
  arg = NULL,
  ...,
  fillcolor = NULL,
  opacity = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
fillcolor	Sets the color filling the active shape' interior. Color: a CSS color string or R color name.
opacity	Sets the opacity of the active shape. Number in the interval [0, 1].
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.

Details

go_layout_activeshape() create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

---

go_layout_coloraxis *Layout coloraxis objects*

Description

Properties of layout.coloraxis.

Usage

```
go_layout_coloraxis(
  arg = NULL,
  ...,
  autocolorscale = NULL,
  cauto = NULL,
  cmax = NULL,
  cmid = NULL,
  cmin = NULL,
  colorbar = NULL,
  colorscale = NULL,
  reversescale = NULL,
  showscale = NULL,
```

```
  skip_invalid = FALSE
)

go_layout_coloraxis_colorbar(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  dtick = NULL,
  exponentformat = NULL,
  labelalias = NULL,
  len = NULL,
  lenmode = NULL,
  minexponent = NULL,
  nticks = NULL,
  orientation = NULL,
  outlinecolor = NULL,
  outlinewidth = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thickness = NULL,
  thicknessmode = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabeloverflow = NULL,
  ticklabelposition = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  x = NULL,
  xanchor = NULL,
  xpad = NULL,
  xref = NULL,
```

```
  y = NULL,
  yanchor = NULL,
  ypad = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_layout_coloraxis_colorbar_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_coloraxis_colorbar_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_layout_coloraxis_colorbar_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_layout_coloraxis_colorbar_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
```

```
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
autocolorscale	Determines whether the colorscale is a default palette (‘autocolorscale: true‘) or the palette determined by ‘colorscale‘. In case ‘colorscale‘ is unspecified or ‘autocolorscale‘ is true, the default palette will be chosen according to whether numbers in the ‘color‘ array are all positive, all negative or mixed. Logical.
cauto	Determines whether or not the color domain is computed with respect to the input data (here corresponding trace color array(s)) or the bounds set in ‘cmin‘ and ‘cmax‘ Defaults to ‘false‘ when ‘cmin‘ and ‘cmax‘ are set by the user. Logical.
cmax	Sets the upper bound of the color domain. Value should have the same units as corresponding trace color array(s) and if set, ‘cmin‘ must be set as well. Number.
cmid	Sets the mid-point of the color domain by scaling ‘cmin‘ and/or ‘cmax‘ to be equidistant to this point. Value should have the same units as corresponding trace color array(s). Has no effect when ‘cauto‘ is ‘false‘. Number.
cmin	Sets the lower bound of the color domain. Value should have the same units as corresponding trace color array(s) and if set, ‘cmax‘ must be set as well. Number.
colorbar	Named list or `go_layout_coloraxis_colorbar()` object.
colorscale	Sets the colorscale. The colorscale must be an array containing arrays mapping a normalized value to an rgb, rgba, hex, hsl, hsla, hwb, or named color string. At minimum, a mapping for the lowest (0) and highest (1) values are required. For example, ‘[[0, ’rgb(0,0,255)’], [1, ’rgb(255,0,0)’]]‘. To control the bounds of the colorscale in color space, use ‘cmin‘ and ‘cmax‘. Alternatively, ‘colorscale‘ may be a palette name string of the following list: Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Rainbow,RdBu,Reds,Vi Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in `px_colors()` (append `"_r"` to reverse).
reversescale	Reverses the color mapping if true. If true, ‘cmin‘ will correspond to the last color in the array and ‘cmax‘ will correspond to the first color. Logical.
showscale	Determines whether or not a colorbar is displayed for this trace. Logical.
skip_invalid	If `TRUE`, silently drop invalid properties instead of raising an error.

`bgcolor` Sets the color of padded area. Color: a CSS color string or R color name.

`bordercolor` Sets the axis line color. Color: a CSS color string or R color name.

`borderwidth` Sets the width (in px) or the border enclosing this color bar. Number in the interval [0, Inf].

`dtick` Sets the step in-between ticks on this color bar. Use with ‘tick0‘. Must be a positive number. Note: unlike on x/y axes, the special "log" and "date" string forms ("L<f>", "D1", "D2", "M<n>") are not supported here: this axis is always linear, so such strings are silently ignored and the step is set to 1. Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: ’USA’, CA: ’Canada’} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`len` Sets the length of the color bar This measure excludes the padding of both ends. That is, the color bar length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this color bar’s length (i.e. the measure in the color variation direction) is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`orientation` Sets the orientation of the colorbar. One of `"h"`, `"v"`.

`outlinecolor` Sets the axis line color. Color: a CSS color string or R color name.

`outlinewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticksuffix Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

thickness Sets the thickness of the color bar This measure excludes the size of the padding, ticks and labels. Number in the interval [0, Inf].

thicknessmode Determines whether this color bar's thickness (i.e. the measure in the constant color direction) is set in units of plot "fraction" or in "pixels". Use 'thickness' to set the value. One of `"fraction"`, `"pixels"`.

tick0 Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the color bar's tick label font Named list or `go_layout_coloraxis_colorbar_tickfont()` object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or `go_layout_coloraxis_colorbar_tickformatstop()` objects.

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". In other cases the default is "hide past div". One of `"allow"`, `"hide past div"`, `"hide past domain"`.

ticklabelposition Determines where tick labels are drawn relative to the ticks. Left and right options are used when 'orientation' is "h", top and bottom when 'orientation' is "v". One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

ticklabelstep Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows

every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode` Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix` Sets a tick label prefix. String.

`ticks` Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_layout_coloraxis_colorbar_title()` object.

`x` Sets the x position with respect to ‘xref‘ of the color bar (in plot fraction). When ‘xref‘ is "paper", defaults to 1.02 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". When ‘xref‘ is "container", defaults to 1 when ‘orientation‘ is "v" and 0.5 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘xref‘ is "container" and between "-2" and 3 if ‘xref‘ is "paper". Number.

`xanchor` Sets this color bar’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the color bar. Defaults to "left" when ‘orientation‘ is "v" and "center" when ‘orientation‘ is "h". One of `"left"`, `"center"`, `"right"`.

`xpad` Sets the amount of padding (in px) along the x direction. Number in the interval [0, Inf].

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

`y` Sets the y position with respect to ‘yref‘ of the color bar (in plot fraction). When ‘yref‘ is "paper", defaults to 0.5 when ‘orientation‘ is "v" and 1.02 when ‘orientation‘ is "h". When ‘yref‘ is "container", defaults to 0.5 when ‘orientation‘ is "v" and 1 when ‘orientation‘ is "h". Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

yanchor — Sets this color bar's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the color bar. Defaults to "middle" when 'orientation' is "v" and "bottom" when 'orientation' is "h". One of "top", "middle", "bottom".

ypad — Sets the amount of padding (in px) along the y direction. Number in the interval [0, Inf].

yref — Sets the container 'y' refers to. "container" spans the entire 'height' of the plot. "paper" refers to the height of the plotting area only. One of "container", "paper".

color — Color: a CSS color string or R color name.

family — HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition — Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow — Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size — Number in the interval [1, Inf].

style — Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase — Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant — Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight — Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

name — When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled:

	false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.
`value`	string - dtickformat for described zoom level, the same as "tickformat" String.
`font`	Sets this color bar’s title font. Named list or `go_layout_coloraxis_colorbar_title_font()` object.
`side`	Determines the location of color bar’s title with respect to the color bar. Defaults to "top" when ‘orientation‘ is "v" and defaults to "right" when ‘orientation‘ is "h". One of `"right"`, `"top"`, `"bottom"`.
`text`	Sets the title of the color bar. String.

Details

`go_layout_coloraxis()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`).

Nested objects documented here:

`go_layout_coloraxis_colorbar()` `layout.coloraxis.colorbar`

`go_layout_coloraxis_colorbar_tickfont()` `layout.coloraxis.colorbar.tickfont`: Sets the color bar’s tick label font

`go_layout_coloraxis_colorbar_tickformatstop()` `layout.coloraxis.colorbar.tickformatstop`

`go_layout_coloraxis_colorbar_title()` `layout.coloraxis.colorbar.title`

`go_layout_coloraxis_colorbar_title_font()` `layout.coloraxis.colorbar.title.font`: Sets this color bar’s title font.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

`go_layout_colorscale` *Layout colorscale objects*

Description

Properties of `layout.colorscale`.

Usage

```
go_layout_colorscale(
  arg = NULL,
  ...,
  diverging = NULL,
  sequential = NULL,
  sequentialminus = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
diverging	Sets the default diverging colorscale. Note that 'autocolorscale' must be true for this attribute to work. Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).
sequential	Sets the default sequential colorscale for positive values. Note that 'autocolorscale' must be true for this attribute to work. Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).
sequentialminus	Sets the default sequential colorscale for negative values. Note that 'autocolorscale' must be true for this attribute to work. Colorscale: a vector of colors, a two-column data.frame of (value, color), or the name of a sequential, diverging or cyclical scale in px_colors() (append "_r" to reverse).
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.

Details

go_layout_colorscale() create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

go_layout_font *Layout font objects*

Description

Sets the global font. Note that fonts used in traces and other layout components inherit from the global font.

Usage

```
go_layout_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
color	Color: a CSS color string or R color name.
family	HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.
lineposition	Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.
shadow	Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.
size	Number in the interval [1, Inf].

style	Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".
textcase	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".
variant	Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
weight	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.

Details

go_layout_font() create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

---

go_layout_geo *Layout geo objects*

Description

Properties of layout.geo.

Usage

```
go_layout_geo(
  arg = NULL,
  ...,
  bgcolor = NULL,
  center = NULL,
  coastlinecolor = NULL,
  coastlinewidth = NULL,
  countrycolor = NULL,
  countrywidth = NULL,
  domain = NULL,
  fitbounds = NULL,
  framecolor = NULL,
  framewidth = NULL,
```

```
  lakecolor = NULL,
  landcolor = NULL,
  lataxis = NULL,
  lonaxis = NULL,
  oceancolor = NULL,
  projection = NULL,
  resolution = NULL,
  rivercolor = NULL,
  riverwidth = NULL,
  scope = NULL,
  showcoastlines = NULL,
  showcountries = NULL,
  showframe = NULL,
  showlakes = NULL,
  showland = NULL,
  showocean = NULL,
  showrivers = NULL,
  showsubunits = NULL,
  subunitcolor = NULL,
  subunitwidth = NULL,
  uirevision = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_layout_geo_center(
  arg = NULL,
  ...,
  lat = NULL,
  lon = NULL,
  skip_invalid = FALSE
)

go_layout_geo_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_layout_geo_lataxis(
  arg = NULL,
  ...,
  dtick = NULL,
  gridcolor = NULL,
```

```
  griddash = NULL,
  gridwidth = NULL,
  range = NULL,
  showgrid = NULL,
  tick0 = NULL,
  skip_invalid = FALSE
)

go_layout_geo_lonaxis(
  arg = NULL,
  ...,
  dtick = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
  range = NULL,
  showgrid = NULL,
  tick0 = NULL,
  skip_invalid = FALSE
)

go_layout_geo_projection(
  arg = NULL,
  ...,
  distance = NULL,
  maxscale = NULL,
  minscale = NULL,
  parallels = NULL,
  rotation = NULL,
  scale = NULL,
  tilt = NULL,
  type = NULL,
  skip_invalid = FALSE
)

go_layout_geo_projection_rotation(
  arg = NULL,
  ...,
  lat = NULL,
  lon = NULL,
  roll = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.

...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
bgcolor	Set the background color of the map Color: a CSS color string or R color name.
center	Named list or go_layout_geo_center() object.
coastlinecolor	Sets the coastline color. Color: a CSS color string or R color name.
coastlinewidth	Sets the coastline stroke width (in px). Number in the interval [0, Inf].
countrycolor	Sets line color of the country boundaries. Color: a CSS color string or R color name.
countrywidth	Sets line width (in px) of the country boundaries. Number in the interval [0, Inf].
domain	Named list or go_layout_geo_domain() object.
fitbounds	Determines if this subplot's view settings are auto-computed to fit trace data. On scoped maps, setting 'fitbounds' leads to 'center.lon' and 'center.lat' getting auto-filled. On maps with a non-clipped projection, setting 'fitbounds' leads to 'center.lon', 'center.lat', and 'projection.rotation.lon' getting auto-filled. On maps with a clipped projection, setting 'fitbounds' leads to 'center.lon', 'center.lat', 'projection.rotation.lon', 'projection.rotation.lat', 'lonaxis.range' and 'lataxis.range' getting auto-filled. If "locations" (default), only the trace's visible locations are considered in the 'fitbounds' computations. If "geojson", the entire trace input 'geojson' (if provided) is considered in the 'fitbounds' computations. If FALSE, the view settings are used as-is; set this to opt out of auto-fitting. One of "locations", "geojson", FALSE.
framecolor	Sets the color the frame. Color: a CSS color string or R color name.
framewidth	Sets the stroke width (in px) of the frame. Number in the interval [0, Inf].
lakecolor	Sets the color of the lakes. Color: a CSS color string or R color name.
landcolor	Sets the land mass color. Color: a CSS color string or R color name.
lataxis	Named list or go_layout_geo_lataxis() object.
lonaxis	Named list or go_layout_geo_lonaxis() object.
oceancolor	Sets the ocean color Color: a CSS color string or R color name.
projection	Named list or go_layout_geo_projection() object.
resolution	Sets the resolution of the base layers. The values have units of km/mm e.g. 110 corresponds to a scale ratio of 1:110,000,000. One of 110, 50.
rivercolor	Sets color of the rivers. Color: a CSS color string or R color name.
riverwidth	Sets the stroke width (in px) of the rivers. Number in the interval [0, Inf].
scope	Set the scope of the map. One of "africa", "antarctica", "asia", "europe", "north america", "oceania", "south america", "usa", "world".
showcoastlines	Sets whether or not the coastlines are drawn. Logical.
showcountries	Sets whether or not country boundaries are drawn. Logical.
showframe	Sets whether or not a frame is drawn around the map. Logical.
showlakes	Sets whether or not lakes are drawn. Logical.

showland	Sets whether or not land masses are filled in color. Logical.
showocean	Sets whether or not oceans are filled in color. Logical.
showrivers	Sets whether or not rivers are drawn. Logical.
showsubunits	Sets whether or not boundaries of subunits within countries (e.g. states, provinces) are drawn. Logical.
subunitcolor	Sets the color of the subunits boundaries. Color: a CSS color string or R color name.
subunitwidth	Sets the stroke width (in px) of the subunits boundaries. Number in the interval [0, Inf].
uirevision	Controls persistence of user-driven changes in the view (projection and center). Defaults to 'layout.uirevision'. Any value.
visible	Sets the default visibility of the base layers. Logical.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
lat	go_layout_geo_center(): Sets the latitude of the map's center. For all projection types, the map's latitude center lies at the middle of the latitude range by default. Number. go_layout_geo_projection_rotation(): Rotates the map along meridians (in degrees North). Number.
lon	go_layout_geo_center(): Sets the longitude of the map's center. By default, the map's longitude center lies at the middle of the longitude range for scoped projection and above 'projection.rotation.lon' otherwise. Number. go_layout_geo_projection_rotation(): Rotates the map along parallels (in degrees East). Defaults to the center of the 'lonaxis.range' values. Number.
column	If there is a layout grid, use the domain for this column in the grid for this geo subplot. Note that geo subplots are constrained by domain. In general, when 'projection.scale' is set to 1. a map will fit either its x or y domain, but not both. Integer in the interval [0, Inf].
row	If there is a layout grid, use the domain for this row in the grid for this geo subplot. Note that geo subplots are constrained by domain. In general, when 'projection.scale' is set to 1. a map will fit either its x or y domain, but not both. Integer in the interval [0, Inf].
x	Sets the horizontal domain of this geo subplot (in plot fraction). Note that geo subplots are constrained by domain. In general, when 'projection.scale' is set to 1. a map will fit either its x or y domain, but not both. Vector or list of 2 element(s).
y	Sets the vertical domain of this geo subplot (in plot fraction). Note that geo subplots are constrained by domain. In general, when 'projection.scale' is set to 1. a map will fit either its x or y domain, but not both. Vector or list of 2 element(s).
dtick	Sets the graticule's longitude/latitude tick step. Number.
gridcolor	Sets the graticule's stroke color. Color: a CSS color string or R color name.
griddash	Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

gridwidth — Sets the graticule's stroke width (in px). Number in the interval [0, Inf].

range — Sets the range of this axis (in degrees), sets the map's clipped coordinates. Vector or list of 2 element(s).

showgrid — Sets whether or not graticule are shown on the map. Logical.

tick0 — Sets the graticule's starting tick longitude/latitude. Number.

distance — For satellite projection type only. Sets the distance from the center of the sphere to the point of view as a proportion of the sphere's radius. Number in the interval [1.001, Inf].

maxscale — Sets the maximum zoom level of the map view, relative to 'projection.scale'. A 'maxscale' of 2 (200%) prevents the user from zooming in beyond twice the base zoom level. Must be greater than or equal to "0"; has no upper bound. Defaults to "null" for no upper bound on zoom. Number in the interval [0, Inf].

minscale — Sets the minimum zoom level of the map view, relative to 'projection.scale'. A 'minscale' of 0.5 (50%) prevents the user from zooming out beyond half the base zoom level. Must be greater than or equal to "0"; has no upper bound. The default of 0 imposes no lower bound on zoom. Number in the interval [0, Inf].

parallels — For conic projection types only. Sets the parallels (tangent, secant) where the cone intersects the sphere. Vector or list of 2 element(s).

rotation — Named list or go_layout_geo_projection_rotation() object.

scale — Zooms in or out on the map view. A scale of 1 corresponds to the largest zoom level that fits the map's lon and lat ranges. Number in the interval [0, Inf].

tilt — For satellite projection type only. Sets the tilt angle of perspective projection. Number.

type — Sets the projection type. One of "airy", "aitoff", "albers", "albers usa", "august", "azimuthal equal area", "azimuthal equidistant", "baker", "bertin1953", "boggs", "bonne", "bottomley", "bromley", "collignon", "conic conformal", "conic equal area", "conic equidistant", "craig", "craster", "cylindrical equal area", "cylindrical stereographic", "eckert1", "eckert2", "eckert3", "eckert4", "eckert5", "eckert6", "eisenlohr", "equal earth", "equirectangular", "fahey", "foucaut", "foucaut sinusoidal", "ginzburg4", "ginzburg5", "ginzburg6", "ginzburg8", "ginzburg9", "gnomonic", "gringorten", "gringorten quincuncial", "guyou", "hammer", "hill", "homolosine", "hufnagel", "hyperelliptical", "kavrayskiy7", "lagrange", "larrivee", "laskowski", "loximuthal", "mercator", "miller", "mollweide", "mt flat polar parabolic", "mt flat polar quartic", "mt flat polar sinusoidal", "natural earth", "natural earth1", "natural earth2", "nell hammer", "nicolosi", "orthographic", "patterson", "peirce quincuncial", "polyconic", "rectangular polyconic", "robinson", "satellite", "sinu mollweide", "sinusoidal", "stereographic", "times", "transverse mercator", "van der grinten", "van der grinten2", "van der grinten3", "van der grinten4", "wagner4", "wagner6", "wiechel", "winkel tripel", "winkel3".

roll — Roll the map (in degrees) For example, a roll of 180 makes the map appear upside down. Number.

Details

go_layout_geo() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Nested objects documented here:

go_layout_geo_center() layout.geo.center

go_layout_geo_domain() layout.geo.domain

go_layout_geo_lataxis() layout.geo.lataxis

go_layout_geo_lonaxis() layout.geo.lonaxis

go_layout_geo_projection() layout.geo.projection

go_layout_geo_projection_rotation() layout.geo.projection.rotation

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

---

go_layout_grid *Layout grid objects*

Description

Properties of layout.grid.

Usage

```
go_layout_grid(
  arg = NULL,
  ...,
  columns = NULL,
  domain = NULL,
  pattern = NULL,
  roworder = NULL,
  rows = NULL,
  subplots = NULL,
  xaxes = NULL,
  xgap = NULL,
  xside = NULL,
  yaxes = NULL,
  ygap = NULL,
```

```
  yside = NULL,
  skip_invalid = FALSE
)

go_layout_grid_domain(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
columns	The number of columns in the grid. If you provide a 2D ‘subplots‘ array, the length of its longest row is used as the default. If you give an ‘xaxes‘ array, its length is used as the default. But it’s also possible to have a different length, if you want to leave a row at the end for non-cartesian subplots. Integer in the interval [1, Inf].
domain	Named list or `go_layout_grid_domain()` object.
pattern	If no ‘subplots‘, ‘xaxes‘, or ‘yaxes‘ are given but we do have ‘rows‘ and ‘columns‘, we can generate defaults using consecutive axis IDs, in two ways: "coupled" gives one x axis per column and one y axis per row. "independent" uses a new xy pair for each cell, left-to-right across each row then iterating rows according to ‘roworder‘. One of `"independent"`, `"coupled"`.
roworder	Is the first row the top or the bottom? Note that columns are always enumerated from left to right. One of `"top to bottom"`, `"bottom to top"`.
rows	The number of rows in the grid. If you provide a 2D ‘subplots‘ array or a ‘yaxes‘ array, its length is used as the default. But it’s also possible to have a different length, if you want to leave a row at the end for non-cartesian subplots. Integer in the interval [1, Inf].
subplots	Used for freeform grids, where some axes may be shared across subplots but others are not. Each entry should be a cartesian subplot id, like "xy" or "x3y2", or ** to leave that cell empty. You may reuse x axes within the same column, and y axes within the same row. Non-cartesian subplots and traces that support ‘domain‘ can place themselves in this grid separately using the ‘gridcell‘ attribute. Vector or list of up to 1 element(s).
xaxes	Used with ‘yaxes‘ when the x and y axes are shared across columns and rows. Each entry should be an x axis id like "x", "x2", etc., or *" to not put an x axis in that column. Entries other than "* must be unique. Ignored if ‘subplots‘ is present. If missing but ‘yaxes‘ is present, will generate consecutive IDs. Vector or list of up to 1 element(s).

xgap	Horizontal space between grid cells, expressed as a fraction of the total width available to one cell. Defaults to 0.1 for coupled-axes grids and 0.2 for independent grids. Number in the interval [0, 1].
xside	Sets where the x axis labels and titles go. "bottom" means the very bottom of the grid. "bottom plot" is the lowest plot that each x axis is used in. "top" and "top plot" are similar. One of `"bottom"`, `"bottom plot"`, `"top plot"`, `"top"`.
yaxes	Used with ‘yaxes‘ when the x and y axes are shared across columns and rows. Each entry should be an y axis id like "y", "y2", etc., or *" to not put a y axis in that row. Entries other than "* must be unique. Ignored if ‘subplots‘ is present. If missing but ‘xaxes‘ is present, will generate consecutive IDs. Vector or list of up to 1 element(s).
ygap	Vertical space between grid cells, expressed as a fraction of the total height available to one cell. Defaults to 0.1 for coupled-axes grids and 0.3 for independent grids. Number in the interval [0, 1].
yside	Sets where the y axis labels and titles go. "left" means the very left edge of the grid. "left plot" is the leftmost plot that each y axis is used in. "right" and "right plot" are similar. One of `"left"`, `"left plot"`, `"right plot"`, `"right"`.
skip_invalid	If `TRUE`, silently drop invalid properties instead of raising an error.
x	Sets the horizontal domain of this grid subplot (in plot fraction). The first and last cells end exactly at the domain edges, with no grout around the edges. Vector or list of 2 element(s).
y	Sets the vertical domain of this grid subplot (in plot fraction). The first and last cells end exactly at the domain edges, with no grout around the edges. Vector or list of 2 element(s).

Details

`go_layout_grid()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`).

Nested objects documented here:

`go_layout_grid_domain()` `layout.grid.domain`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

go_layout_hoverlabel *Layout hoverlabel objects*

Description

Properties of `layout.hoverlabel`.

Usage

```
go_layout_hoverlabel(
  arg = NULL,
  ...,
  align = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
  grouptitlefont = NULL,
  namelength = NULL,
  showarrow = NULL,
  skip_invalid = FALSE
)

go_layout_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_hoverlabel_grouptitlefont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
```

```
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
align	Sets the horizontal alignment of the text content within hover label box. Has an effect only if the hover label text spans two or more lines One of `"left"`, `"right"`, `"auto"`.
bgcolor	Sets the background color of all hover labels on graph Color: a CSS color string or R color name.
bordercolor	Sets the border color of all hover labels on graph. Color: a CSS color string or R color name.
font	Sets the default hover label font used by all traces on the graph. Named list or `go_layout_hoverlabel_font()` object.
grouptitlefont	Sets the font for group titles in hover (unified modes). Defaults to 'hoverlabel.font'. Named list or `go_layout_hoverlabel_grouptitlefont()` object.
namelength	Sets the default length (in number of characters) of the trace name in the hover labels for all traces. -1 shows the whole name regardless of length. 0-3 shows the first 0-3 characters, and an integer >3 will show the whole name if it is less than that many characters, but if it is longer, will truncate to 'namelength - 3' characters and add an ellipsis. Integer in the interval [-1, Inf].
showarrow	Sets whether or not to show the hover label arrow/triangle pointing to the data point. Logical.
skip_invalid	If `TRUE`, silently drop invalid properties instead of raising an error.
color	Color: a CSS color string or R color name.
family	HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.
lineposition	Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.
shadow	Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.
size	Number in the interval [1, Inf].
style	Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

textcase	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".
variant	Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
weight	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

Details

go_layout_hoverlabel() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Nested objects documented here:

go_layout_hoverlabel_font() layout.hoverlabel.font: Sets the default hover label font used by all traces on the graph.

go_layout_hoverlabel_grouptitlefont() layout.hoverlabel.grouptitlefont: Sets the font for group titles in hover (unified modes). Defaults to 'hoverlabel.font'.

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

---

go_layout_legend *Layout legend objects*

Description

Properties of layout.legend.

Usage

```
go_layout_legend(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  entrywidth = NULL,
  entrywidthmode = NULL,
  font = NULL,
  groupclick = NULL,
```

```
  groupdoubleclick = NULL,
  grouptitlefont = NULL,
  indentation = NULL,
  itemclick = NULL,
  itemdoubleclick = NULL,
  itemheight = NULL,
  itemsizing = NULL,
  itemwidth = NULL,
  maxheight = NULL,
  orientation = NULL,
  title = NULL,
  titleclick = NULL,
  titledoubleclick = NULL,
  tracegroupgap = NULL,
  traceorder = NULL,
  uirevision = NULL,
  valign = NULL,
  visible = NULL,
  x = NULL,
  xanchor = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_layout_legend_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_legend_grouptitlefont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
```

```
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_legend_title(
  arg = NULL,
  ...,
  font = NULL,
  side = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_layout_legend_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
bgcolor	Sets the legend background color. Defaults to ‘layout.paper_bgcolor‘. Color: a CSS color string or R color name.
bordercolor	Sets the color of the border enclosing the legend. Color: a CSS color string or R color name.
borderwidth	Sets the width (in px) of the border enclosing the legend. Number in the interval [0, Inf].
entrywidth	Sets the width (in px or fraction) of the legend. Use 0 to size the entry based on

the text width, when ‘entrywidthmode‘ is set to "pixels". Number in the interval [0, Inf].

`entrywidthmode` Determines what entrywidth means. One of `"fraction"`, `"pixels"`.

`font` `go_layout_legend()`: Sets the font used to text the legend items. Named list or `go_layout_legend_font()` object.

`go_layout_legend_title()`: Sets this legend's title font. Defaults to ‘legend.font‘ with its size increased about 20%. Named list or `go_layout_legend_title_font()` object.

`groupclick` Determines the behavior on legend group item click. "toggleitem" toggles the visibility of the individual item clicked on the graph. "togglegroup" toggles the visibility of all items in the same legendgroup as the item clicked on the graph. One of `"toggleitem"`, `"togglegroup"`.

`groupdoubleclick`

Determines the behavior on legend group item double-click. "toggleitem" toggles the visibility of the individual item clicked on the graph. "togglegroup" toggles the visibility of all items in the same legendgroup as the item clicked on the graph. Defaults to the value of ‘groupclick‘. One of `"toggleitem"`, `"togglegroup"`.

`grouptitlefont` Sets the font for group titles in legend. Defaults to ‘legend.font‘ with its size increased about 10%. Named list or `go_layout_legend_grouptitlefont()` object.

`indentation` Sets the indentation (in px) of the legend entries. Number in the interval [-15, Inf].

`itemclick` Determines the behavior on legend item click. "toggle" toggles the visibility of the item clicked on the graph. "toggleothers" makes the clicked item the sole visible item on the graph. FALSE disables legend item click interactions. One of `"toggle"`, `"toggleothers"`, `FALSE`.

`itemdoubleclick`

Determines the behavior on legend item double-click. "toggle" toggles the visibility of the item clicked on the graph. "toggleothers" makes the clicked item the sole visible item on the graph. FALSE disables legend item double-click interactions. One of `"toggle"`, `"toggleothers"`, `FALSE`.

`itemheight` Sets the height (in px) of the legend item fill swatch. A taller swatch shows more of a trace "fill" or "fillpattern". Only traces with a "fill" use this value. Every legend item grows to fit, so a larger value makes every row taller. Number in the interval [6, Inf].

`itemsizing` Determines if the legend items symbols scale with their corresponding "trace" attributes or remain "constant" independent of the symbol size on the graph. One of `"trace"`, `"constant"`.

`itemwidth` Sets the width (in px) of the legend item symbols (the part other than the title.text). Number in the interval [30, Inf].

`maxheight` Sets the max height (in px) of the legend, or max height ratio (reference height * ratio) if less than or equal to 1. Default value is: 0.5 for horizontal legends; 1 for vertical legends. The minimum allowed height is 30px. For a ratio of 0.5, the legend will take up to 50% of the reference height before displaying

a scrollbar. The reference height is the full layout height with the following exception: vertically oriented legends with a ‘yref‘ of ‘"paper", located to the side of the plot. In this case, the reference height is the plot height. Number in the interval [0, Inf].

`orientation` Sets the orientation of the legend. One of `"v"`, `"h"`.

`title` String (the title text), named list or `go_layout_legend_title()` object.

`titleclick` Determines the behavior on legend title click. "toggle" toggles the visibility of all items in the legend. "toggleothers" toggles the visibility of all other legends. FALSE disables legend title click interactions. Defaults to "toggle" when there are multiple legends, FALSE otherwise. Not supported for legends containing pie and pie-like traces. One of `"toggle"`, `"toggleothers"`, `FALSE`.

`titledoubleclick` Determines the behavior on legend title double-click. "toggle" toggles the visibility of all items in the legend. "toggleothers" toggles the visibility of all other legends. FALSE disables legend title double-click interactions. Defaults to "toggleothers" when there are multiple legends, FALSE otherwise. Not supported for legends containing pie and pie-like traces. One of `"toggle"`, `"toggleothers"`, `FALSE`.

`tracegroupgap` Sets the amount of vertical space (in px) between legend groups. Number in the interval [0, Inf].

`traceorder` Determines the order at which the legend items are displayed. If "normal", the items are displayed top-to-bottom in the same order as the input data. If "reversed", the items are displayed in the opposite order as "normal". If "grouped", the items are displayed in groups (when a trace ‘legendgroup‘ is provided). if "grouped+reversed", the items are displayed in the opposite order as "grouped". Any combination of `"reversed"`, `"grouped"` joined with `"+"`, or one of `"normal"`.

`uirevision` Controls persistence of legend-driven changes in trace and pie label visibility. Defaults to ‘layout.uirevision‘. Any value.

`valign` Sets the vertical alignment of the symbols with respect to their associated text. One of `"top"`, `"middle"`, `"bottom"`.

`visible` Determines whether or not this legend is visible. Logical.

`x` Sets the x position with respect to ‘xref‘ (in normalized coordinates) of the legend. When ‘xref‘ is "paper", defaults to 1.02 for vertical legends and defaults to 0 for horizontal legends. When ‘xref‘ is "container", defaults to 1 for vertical legends and defaults to 0 for horizontal legends. Must be between 0 and 1 if ‘xref‘ is "container". and between "-2" and 3 if ‘xref‘ is "paper". Number.

`xanchor` Sets the legend’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the legend. Value "auto" anchors legends to the right for ‘x‘ values greater than or equal to 2/3, anchors legends to the left for ‘x‘ values less than or equal to 1/3 and anchors legends with respect to their center otherwise. One of `"auto"`, `"left"`, `"center"`, `"right"`.

`xref` Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.

y — Sets the y position with respect to ‘yref‘ (in normalized coordinates) of the legend. When ‘yref‘ is "paper", defaults to 1 for vertical legends, defaults to "-0.1" for horizontal legends on graphs w/o range sliders and defaults to 1.1 for horizontal legends on graph with one or multiple range sliders. When ‘yref‘ is "container", defaults to 1. Must be between 0 and 1 if ‘yref‘ is "container" and between "-2" and 3 if ‘yref‘ is "paper". Number.

yanchor — Sets the legend's vertical position anchor. This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the legend. Value "auto" anchors legends at their bottom for ‘y‘ values less than or equal to 1/3, anchors legends to at their top for ‘y‘ values greater than or equal to 2/3 and anchors legends with respect to their middle otherwise. One of `"auto"`, `"top"`, `"middle"`, `"bottom"`.

yref — Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.

skip_invalid — If `TRUE`, silently drop invalid properties instead of raising an error.

color — Color: a CSS color string or R color name.

family — HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition — Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

shadow — Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size — Number in the interval [1, Inf].

style — Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

textcase — Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.

variant — Sets the variant of the font. One of `"normal"`, `"small-caps"`, `"all-small-caps"`, `"all-petite-caps"`, `"petite-caps"`, `"unicase"`.

weight — Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

side — Determines the location of legend's title with respect to the legend items. Defaulted to "top" with ‘orientation‘ is "h". Defaulted to "left" with ‘orientation‘ is "v". The "top left" options could be used to expand top center and top right are for horizontal alignment legend area in both x and y sides. One of `"top"`, `"left"`, `"top left"`, `"top center"`, `"top right"`.

text — Sets the title of the legend. String.

Details

`go_layout_legend()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`).

Nested objects documented here:

`go_layout_legend_font()` `layout.legend.font`: Sets the font used to text the legend items.

`go_layout_legend_grouptitlefont()` `layout.legend.grouptitlefont`: Sets the font for group titles in legend. Defaults to 'legend.font' with its size increased about 10%.

`go_layout_legend_title()` `layout.legend.title`

`go_layout_legend_title_font()` `layout.legend.title.font`: Sets this legend's title font. Defaults to 'legend.font' with its size increased about 20%.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

`go_layout_map` *Layout map objects*

Description

Properties of `layout.map`.

Usage

```
go_layout_map(
  arg = NULL,
  ...,
  bearing = NULL,
  bounds = NULL,
  center = NULL,
  domain = NULL,
  fitbounds = NULL,
  layers = NULL,
  pitch = NULL,
  style = NULL,
  uirevision = NULL,
  zoom = NULL,
  skip_invalid = FALSE
)
```

```
go_layout_map_bounds(
  arg = NULL,
  ...,
  east = NULL,
  north = NULL,
  south = NULL,
  west = NULL,
  skip_invalid = FALSE
)

go_layout_map_center(
  arg = NULL,
  ...,
  lat = NULL,
  lon = NULL,
  skip_invalid = FALSE
)

go_layout_map_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_layout_map_layer(
  arg = NULL,
  ...,
  below = NULL,
  circle = NULL,
  color = NULL,
  coordinates = NULL,
  fill = NULL,
  line = NULL,
  maxzoom = NULL,
  minzoom = NULL,
  name = NULL,
  opacity = NULL,
  source = NULL,
  sourceattribution = NULL,
  sourcelayer = NULL,
  sourcetype = NULL,
  symbol = NULL,
  templateitemname = NULL,
```

```
  type = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_layout_map_layer_circle(
  arg = NULL,
  ...,
  radius = NULL,
  skip_invalid = FALSE
)

go_layout_map_layer_fill(
  arg = NULL,
  ...,
  outlinecolor = NULL,
  skip_invalid = FALSE
)

go_layout_map_layer_line(
  arg = NULL,
  ...,
  dash = NULL,
  width = NULL,
  skip_invalid = FALSE
)

go_layout_map_layer_symbol(
  arg = NULL,
  ...,
  icon = NULL,
  iconsize = NULL,
  placement = NULL,
  text = NULL,
  textfont = NULL,
  textposition = NULL,
  skip_invalid = FALSE
)

go_layout_map_layer_symbol_textfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  size = NULL,
  style = NULL,
  weight = NULL,
  skip_invalid = FALSE
```

)

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
bearing	Sets the bearing angle of the map in degrees counter-clockwise from North (map.bearing). Number.
bounds	Named list or go_layout_map_bounds() object.
center	Named list or go_layout_map_center() object.
domain	Named list or go_layout_map_domain() object.
fitbounds	Determines if this subplot's view settings are auto-computed to fit trace data. If "locations" (default), the view is auto-fit to the lon/lat coordinates of the visible trace data. If FALSE, the view settings are used as-is; set this to opt out of auto-fitting. If 'fitbounds' is enabled but a user provides 'center' or 'zoom', auto-fit will be skipped. One of "locations", FALSE.
layers	List of named lists or go_layout_map_layer() objects.
pitch	Sets the pitch angle of the map (in degrees, where 0 means perpendicular to the surface of the map) (map.pitch). Number.
style	go_layout_map(): Defines the map layers that are rendered by default below the trace layers defined in 'data', which are themselves by default rendered below the layers defined in 'layout.map.layers'. These layers can be defined either explicitly as a Map Style object which can contain multiple layer definitions that load data from any public or private Tile Map Service (TMS or XYZ) or Web Map Service (WMS) or implicitly by using one of the built-in style objects which use WMSes or by using a custom style URL Map Style objects are of the form described in the MapLibre GL JS documentation available at https://maplibre.org/maplibre-style-spec/ The built-in plotly.js styles objects are: basic, carto-darkmatter, carto-darkmatter-nolabels, carto-positron, carto-positron-nolabels, carto-voyager, carto-voyager-nolabels, dark, light, open-street-map, outdoors, satellite, satellite-streets, streets, white-bg. Any value. go_layout_map_layer_symbol_textfont(): Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".
uirevision	Controls persistence of user-driven changes in the view: 'center', 'zoom', 'bearing', 'pitch'. Defaults to 'layout.uirevision'. Any value.
zoom	Sets the zoom level of the map (map.zoom). Number.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
east	Sets the maximum longitude of the map (in degrees East) if 'west', 'south' and 'north' are declared. Number.
north	Sets the maximum latitude of the map (in degrees North) if 'east', 'west' and 'south' are declared. Number.

south — Sets the minimum latitude of the map (in degrees North) if ‘east‘, ‘west‘ and ‘north‘ are declared. Number.

west — Sets the minimum longitude of the map (in degrees East) if ‘east‘, ‘south‘ and ‘north‘ are declared. Number.

lat — Sets the latitude of the center of the map (in degrees North). Number.

lon — Sets the longitude of the center of the map (in degrees East). Number.

column — If there is a layout grid, use the domain for this column in the grid for this map subplot. Integer in the interval [0, Inf].

row — If there is a layout grid, use the domain for this row in the grid for this map subplot. Integer in the interval [0, Inf].

x — Sets the horizontal domain of this map subplot (in plot fraction). Vector or list of 2 element(s).

y — Sets the vertical domain of this map subplot (in plot fraction). Vector or list of 2 element(s).

below — Determines if the layer will be inserted before the layer with the specified ID. If omitted or set to ”, the layer will be inserted above every existing layer. String.

circle — Named list or go_layout_map_layer_circle() object.

color — go_layout_map_layer(): Sets the primary layer color. If ‘type‘ is "circle", color corresponds to the circle color (map.layer.paint.circle-color) If ‘type‘ is "line", color corresponds to the line color (map.layer.paint.line-color) If ‘type‘ is "fill", color corresponds to the fill color (map.layer.paint.fill-color) If ‘type‘ is "symbol", color corresponds to the icon color (map.layer.paint.icon-color) Color: a CSS color string or R color name.

go_layout_map_layer_symbol_textfont(): Color: a CSS color string or R color name.

coordinates — Sets the coordinates array contains [longitude, latitude] pairs for the image corners listed in clockwise order: top left, top right, bottom right, bottom left. Only has an effect for "image" ‘sourcetype‘. Any value.

fill — Named list or go_layout_map_layer_fill() object.

line — Named list or go_layout_map_layer_line() object.

maxzoom — Sets the maximum zoom level (map.layer.maxzoom). At zoom levels equal to or greater than the maxzoom, the layer will be hidden. Number in the interval [0, 24].

minzoom — Sets the minimum zoom level (map.layer.minzoom). At zoom levels less than the minzoom, the layer will be hidden. Number in the interval [0, 24].

name — When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

opacity — Sets the opacity of the layer. If ‘type‘ is "circle", opacity corresponds to the circle opacity (map.layer.paint.circle-opacity) If ‘type‘ is "line", opacity corresponds to the line opacity (map.layer.paint.line-opacity) If ‘type‘ is "fill", opac-

ity corresponds to the fill opacity (map.layer.paint.fill-opacity) If ‘type‘ is "symbol", opacity corresponds to the icon/text opacity (map.layer.paint.text-opacity) Number in the interval [0, 1].

`source` Sets the source data for this layer (map.layer.source). When ‘sourcetype‘ is set to "geojson", ‘source‘ can be a URL to a GeoJSON or a GeoJSON object. When ‘sourcetype‘ is set to "vector" or "raster", ‘source‘ can be a URL or an array of tile URLs. When ‘sourcetype‘ is set to "image", ‘source‘ can be a URL to an image. Any value.

`sourceattribution`
Sets the attribution for this source. String.

`sourcelayer` Specifies the layer to use from a vector tile source (map.layer.source-layer). Required for "vector" source type that supports multiple layers. String.

`sourcetype` Sets the source type for this layer, that is the type of the layer data. One of `"geojson"`, `"vector"`, `"raster"`, `"image"`.

`symbol` Named list or `go_layout_map_layer_symbol()` object.

`templateitemname`
Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`type` Sets the layer type, that is the how the layer data set in ‘source‘ will be rendered With ‘sourcetype‘ set to "geojson", the following values are allowed: "circle", "line", "fill" and "symbol". but note that "line" and "fill" are not compatible with Point GeoJSON geometries. With ‘sourcetype‘ set to "vector", the following values are allowed: "circle", "line", "fill" and "symbol". With ‘sourcetype‘ set to "raster" or "image", only the "raster" value is allowed. One of `"circle"`, `"line"`, `"fill"`, `"symbol"`, `"raster"`.

`visible` Determines whether this layer is displayed Logical.

`radius` Sets the circle radius (map.layer.paint.circle-radius). Has an effect only when ‘type‘ is set to "circle". Number.

`outlinecolor` Sets the fill outline color (map.layer.paint.fill-outline-color). Has an effect only when ‘type‘ is set to "fill". Color: a CSS color string or R color name.

`dash` Sets the length of dashes and gaps (map.layer.paint.line-dasharray). Has an effect only when ‘type‘ is set to "line". Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`width` Sets the line width (map.layer.paint.line-width). Has an effect only when ‘type‘ is set to "line". Number.

`icon` Sets the symbol icon image (map.layer.layout.icon-image). Full list: https://www.mapbox.com/maki-icons/ String.

`iconsize` Sets the symbol icon size (map.layer.layout.icon-size). Has an effect only when ‘type‘ is set to "symbol". Number.

placement — Sets the symbol and/or text placement (map.layer.layout.symbol-placement). If ‘placement‘ is "point", the label is placed where the geometry is located If ‘placement‘ is "line", the label is placed along the line of the geometry If ‘placement‘ is "line-center", the label is placed on the center of the geometry One of "point", "line", "line-center".

text — Sets the symbol text (map.layer.layout.text-field). String.

textfont — Sets the icon text font (color=map.layer.paint.text-color, size=map.layer.layout.text-size). Has an effect only when ‘type‘ is set to "symbol". Named list or go_layout_map_layer_symbol_te object.

textposition — Sets the positions of the ‘text‘ elements with respects to the (x,y) coordinates. One of "top left", "top center", "top right", "middle left", "middle center", "middle right", "bottom left", "bottom center", "bottom right".

family — HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String.

size — Number in the interval [1, Inf].

weight — Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

Details

go_layout_map() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Nested objects documented here:

go_layout_map_bounds() layout.map.bounds

go_layout_map_center() layout.map.center

go_layout_map_domain() layout.map.domain

go_layout_map_layer() layout.map.layer

go_layout_map_layer_circle() layout.map.layer.circle

go_layout_map_layer_fill() layout.map.layer.fill

go_layout_map_layer_line() layout.map.layer.line

go_layout_map_layer_symbol() layout.map.layer.symbol

go_layout_map_layer_symbol_textfont() layout.map.layer.symbol.textfont: Sets the icon text font (color=map.layer.paint.text-color, size=map.layer.layout.text-size). Has an effect only when ‘type‘ is set to "symbol".

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

go_layout_margin *Layout margin objects*

Description

Properties of `layout.margin`.

Usage

```
go_layout_margin(
  arg = NULL,
  ...,
  autoexpand = NULL,
  b = NULL,
  l = NULL,
  pad = NULL,
  r = NULL,
  t = NULL,
  skip_invalid = FALSE
)
```

Arguments

`arg`	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
`...`	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
`autoexpand`	Turns on/off margin expansion computations. Legends, colorbars, updatemenus, sliders, axis rangeselector and rangeslider are allowed to push the margins by defaults. Logical.
`b`	Sets the bottom margin (in px). Number in the interval [0, Inf].
`l`	Sets the left margin (in px). Number in the interval [0, Inf].
`pad`	Sets the amount of padding (in px) between the plotting area and the axis lines Number in the interval [0, Inf].
`r`	Sets the right margin (in px). Number in the interval [0, Inf].
`t`	Sets the top margin (in px). Number in the interval [0, Inf].
`skip_invalid`	If `TRUE`, silently drop invalid properties instead of raising an error.

Details

`go_layout_margin()` create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`).

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

`go_layout_modebar` *Layout modebar objects*

Description

Properties of `layout.modebar`.

Usage

```
go_layout_modebar(
  arg = NULL,
  ...,
  activecolor = NULL,
  add = NULL,
  bgcolor = NULL,
  color = NULL,
  orientation = NULL,
  remove = NULL,
  uirevision = NULL,
  skip_invalid = FALSE
)
```

Arguments

`arg`	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
`...`	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
`activecolor`	Sets the color of the active or hovered on icons in the modebar. Color: a CSS color string or R color name.
`add`	Determines which predefined modebar buttons to add. Please note that these buttons will only be shown if they are compatible with all trace types used in a graph. Similar to ‘config.modeBarButtonsToAdd‘ option. This may include "v1hovermode", "hoverclosest", "hovercompare", "togglehover", "togglespikelines", "drawline", "drawopenpath", "drawclosedpath", "drawcircle", "drawrect", "eraseshape", "downloadJson". String, or a vector of these.

bgcolor	Sets the background color of the modebar. Color: a CSS color string or R color name.
color	Sets the color of the icons in the modebar. Color: a CSS color string or R color name.
orientation	Sets the orientation of the modebar. One of "v", "h".
remove	Determines which predefined modebar buttons to remove. Similar to 'config.modeBarButtonsToRemove' option. This may include "autoScale2d", "autoscale", "hoverCompareCartesian", "hovercompare", "lasso", "lasso2d", "orbitRotation", "orbitrotation", "pan", "pan2d", "pan3d", "reset", "resetCameraDefault3d", "resetCameraLastSave3d", "resetGeo", "resetSankeyGroup", "resetScale2d", "resetViewMap", "resetViews", "resetcameradefault", "resetcameralastsave", "resetsankeygroup", "resetscale", "resetview", "resetviews", "select", "select2d", "sendChartToCloud", "sendcharttocloud", "tableRotation", "tablerotation", "toImage", "toggleHover", "toggleSpikelines", "togglehover", "togglespikelines", "toimage", "zoom", "zoom2d", "zoom3d", "zoomIn2d", "zoomInGeo", "zoomInMap", "zoomOut2d", "zoomOutGeo", "zoomOutMap", "zoomin", "zoomout". String, or a vector of these.
uirevision	Controls persistence of user-driven changes related to the modebar, including 'hovermode', 'dragmode', and 'showspikes' at both the root level and inside subplots. Defaults to 'layout.uirevision'. Any value.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.

Details

go_layout_modebar() create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

---

go_layout_newselection

Layout newselection objects

Description

Properties of layout.newselection.

Usage

```
go_layout_newselection(
  arg = NULL,
  ...,
  line = NULL,
  mode = NULL,
  skip_invalid = FALSE
)

go_layout_newselection_line(
  arg = NULL,
  ...,
  color = NULL,
  dash = NULL,
  width = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
line	Named list or `go_layout_newselection_line()` object.
mode	Describes how a new selection is created. If 'immediate', a new selection is created after first mouse up. If 'gradual', a new selection is not created after first mouse. By adding to and subtracting from the initial selection, this option allows declaring extra outlines of the selection. One of `"immediate"`, `"gradual"`.
skip_invalid	If `TRUE`, silently drop invalid properties instead of raising an error.
color	Sets the line color. By default uses either dark grey or white to increase contrast with background color. Color: a CSS color string or R color name.
dash	Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.
width	Sets the line width (in px). Number in the interval [1, Inf].

Details

`go_layout_newselection()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`).

Nested objects documented here:

`go_layout_newselection_line()` `layout.newselection.line`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

`go_layout_newshape` *Layout newshape objects*

Description

Properties of `layout.newshape`.

Usage

```
go_layout_newshape(
  arg = NULL,
  ...,
  drawdirection = NULL,
  fillcolor = NULL,
  fillrule = NULL,
  label = NULL,
  layer = NULL,
  legend = NULL,
  legendgroup = NULL,
  legendgrouptitle = NULL,
  legendrank = NULL,
  legendwidth = NULL,
  line = NULL,
  name = NULL,
  opacity = NULL,
  showlegend = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_layout_newshape_label(
  arg = NULL,
  ...,
  font = NULL,
  padding = NULL,
  text = NULL,
  textangle = NULL,
  textposition = NULL,
  texttemplate = NULL,
```

```
  texttemplatefallback = NULL,
  xanchor = NULL,
  yanchor = NULL,
  skip_invalid = FALSE
)

go_layout_newshape_label_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_newshape_legendgrouptitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_layout_newshape_legendgrouptitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_newshape_line(
  arg = NULL,
  ...,
  color = NULL,
```

```
  dash = NULL,
  width = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
drawdirection	When 'dragmode' is set to "drawrect", "drawline" or "drawcircle" this limits the drag to be horizontal, vertical or diagonal. Using "diagonal" there is no limit e.g. in drawing lines in any direction. "ortho" limits the draw to be either horizontal or vertical. "horizontal" allows horizontal extend. "vertical" allows vertical extend. One of "ortho", "horizontal", "vertical", "diagonal".
fillcolor	Sets the color filling new shapes' interior. Please note that if using a fillcolor with alpha greater than half, drag inside the active shape starts moving the shape underneath, otherwise a new shape could be started over. Color: a CSS color string or R color name.
fillrule	Determines the path's interior. For more info please visit https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/fill-rule One of "evenodd", "nonzero".
label	Named list or go_layout_newshape_label() object.
layer	Specifies whether new shapes are drawn below gridlines ("below"), between gridlines and traces ("between") or above traces ("above"). One of "below", "above", "between".
legend	Sets the reference to a legend to show new shape in. References to these legends are "legend", "legend2", "legend3", etc. Settings for these legends are set in the layout, under 'layout.legend', 'layout.legend2', etc. Subplot id: "legend", "legend2", "legend3", ...
legendgroup	Sets the legend group for new shape. Traces and shapes part of the same legend group hide/show at the same time when toggling legend items. String.
legendgrouptitle	Named list or go_layout_newshape_legendgrouptitle() object.
legendrank	Sets the legend rank for new shape. Items and groups with smaller ranks are presented on top/left side while with "reversed" 'legend.traceorder' they are on bottom/right side. The default legendrank is 1000, so that you can use ranks less than 1000 to place certain items before all unranked items, and ranks greater than 1000 to go after all unranked items. Number.
legendwidth	Sets the width (in px or fraction) of the legend for new shape. Number in the interval [0, Inf].
line	Named list or go_layout_newshape_line() object.
name	Sets new shape name. The name appears as the legend item. String.
opacity	Sets the opacity of new shapes. Number in the interval [0, 1].

showlegend — Determines whether or not new shape is shown in the legend. Logical.

visible — Determines whether or not new shape is visible. If "legendonly", the shape is not drawn, but can appear as a legend item (provided that the legend itself is visible). One of "legendonly", TRUE, FALSE.

skip_invalid — If TRUE, silently drop invalid properties instead of raising an error.

font — go_layout_newshape_label(): Sets the new shape label text font. Named list or go_layout_newshape_label_font() object.

go_layout_newshape_legendgrouptitle(): Sets this legend group's title font. Named list or go_layout_newshape_legendgrouptitle_font() object.

padding — Sets padding (in px) between edge of label and edge of new shape. Number in the interval [0, Inf].

text — go_layout_newshape_label(): Sets the text to display with the new shape. It is also used for legend item if 'name' is not provided. String.

go_layout_newshape_legendgrouptitle(): Sets the title of the legend group. String.

textangle — Sets the angle at which the label text is drawn with respect to the horizontal. For lines, angle "auto" is the same angle as the line. For all other shapes, angle "auto" is horizontal. Angle in degrees.

textposition — Sets the position of the label text relative to the new shape. Supported values for rectangles, circles and paths are "top left", "top center", "top right", "middle left", "middle center", "middle right", "bottom left", "bottom center", and "bottom right". Supported values for lines are "start", "middle", and "end". Default: "middle center" for rectangles, circles, and paths; "middle" for lines. One of "top left", "top center", "top right", "middle left", "middle center", "middle right", "bottom left", "bottom center", "bottom right", "start", "middle", "end".

texttemplate — Template string used for rendering the new shape's label. Note that this will override 'text'. Variables are inserted using %{variable}, for example "x0: %{x0}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{x0:$.2f}". See https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{x0|%m %b %Y}". See https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. A single multiplication or division operation may be applied to numeric variables, and combined with d3 number formatting, for example "Length in cm: %{x0"2.54}", "%{slope"60:.1f} meters per second." For log axes, variable values are given in log units. For date axes, x/y coordinate variables and center variables use datetimes, while all other variable values use values in ms. Finally, the template string has access to variables 'x0', 'x1', 'y0', 'y1', 'slope', 'dx', 'dy', 'width', 'height', 'length', 'xcenter', 'ycenter', 'simpleXVariables' and 'simpleYVariables'. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. String.

texttemplatefallback
: Fallback string that's displayed when a variable referenced in a template is missing. If the boolean value 'false' is passed in, the specifier with the missing variable will be displayed. Any value.

xanchor
: Sets the label's horizontal position anchor This anchor binds the specified 'textposition' to the "left", "center" or "right" of the label text. For example, if 'textposition' is set to "top right" and 'xanchor' to "right" then the right-most portion of the label text lines up with the right-most edge of the new shape. One of `"auto"`, `"left"`, `"center"`, `"right"`.

yanchor
: Sets the label's vertical position anchor This anchor binds the specified 'textposition' to the "top", "middle" or "bottom" of the label text. For example, if 'textposition' is set to "top right" and 'yanchor' to "top" then the top-most portion of the label text lines up with the top-most edge of the new shape. One of `"top"`, `"middle"`, `"bottom"`.

color
: `go_layout_newshape_label_font()`, `go_layout_newshape_legendgrouptitle_font()`: Color: a CSS color string or R color name.

  `go_layout_newshape_line()`: Sets the line color. By default uses either dark grey or white to increase contrast with background color. Color: a CSS color string or R color name.

family
: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

shadow
: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: Number in the interval [1, Inf].

style
: Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

textcase
: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.

variant
: Sets the variant of the font. One of `"normal"`, `"small-caps"`, `"all-small-caps"`, `"all-petite-caps"`, `"petite-caps"`, `"unicase"`.

weight
: Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

dash
: Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

width
: Sets the line width (in px). Number in the interval [0, Inf].

Details

`go_layout_newshape()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`).

Nested objects documented here:

`go_layout_newshape_label()` `layout.newshape.label`

`go_layout_newshape_label_font()` `layout.newshape.label.font`: Sets the new shape label text font.

`go_layout_newshape_legendgrouptitle()` `layout.newshape.legendgrouptitle`

`go_layout_newshape_legendgrouptitle_font()` `layout.newshape.legendgrouptitle.font`: Sets this legend group's title font.

`go_layout_newshape_line()` `layout.newshape.line`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

`go_layout_polar` *Layout polar objects*

Description

Properties of `layout.polar`.

Usage

```
go_layout_polar(
  arg = NULL,
  ...,
  angularaxis = NULL,
  bargap = NULL,
  barmode = NULL,
  bgcolor = NULL,
  domain = NULL,
  gridshape = NULL,
  hole = NULL,
  radialaxis = NULL,
  sector = NULL,
  uirevision = NULL,
  skip_invalid = FALSE
```

```
)

go_layout_polar_angularaxis(
  arg = NULL,
  ...,
  autotypenumbers = NULL,
  categoryarray = NULL,
  categoryorder = NULL,
  color = NULL,
  direction = NULL,
  dtick = NULL,
  exponentformat = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
  hoverformat = NULL,
  labelalias = NULL,
  layer = NULL,
  linecolor = NULL,
  linewidth = NULL,
  minexponent = NULL,
  minorloglabels = NULL,
  nticks = NULL,
  period = NULL,
  rotation = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showgrid = NULL,
  showline = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  thetaunit = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
```

```
  type = NULL,
  uirevision = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_layout_polar_angularaxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_polar_angularaxis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_layout_polar_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_layout_polar_radialaxis(
  arg = NULL,
  ...,
  angle = NULL,
  autorange = NULL,
  autorangeoptions = NULL,
```

```
autotickangles = NULL,
autotypenumbers = NULL,
calendar = NULL,
categoryarray = NULL,
categoryorder = NULL,
color = NULL,
dtick = NULL,
exponentformat = NULL,
gridcolor = NULL,
griddash = NULL,
gridwidth = NULL,
hoverformat = NULL,
labelalias = NULL,
layer = NULL,
linecolor = NULL,
linewidth = NULL,
maxallowed = NULL,
minallowed = NULL,
minexponent = NULL,
minorloglabels = NULL,
nticks = NULL,
range = NULL,
rangemode = NULL,
separatethousands = NULL,
showexponent = NULL,
showgrid = NULL,
showline = NULL,
showticklabels = NULL,
showtickprefix = NULL,
showticksuffix = NULL,
side = NULL,
tick0 = NULL,
tickangle = NULL,
tickcolor = NULL,
tickfont = NULL,
tickformat = NULL,
tickformatstops = NULL,
ticklabelstep = NULL,
ticklen = NULL,
tickmode = NULL,
tickprefix = NULL,
ticks = NULL,
ticksuffix = NULL,
ticktext = NULL,
tickvals = NULL,
tickwidth = NULL,
title = NULL,
type = NULL,
```

```
  uirevision = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_layout_polar_radialaxis_autorangeoptions(
  arg = NULL,
  ...,
  clipmax = NULL,
  clipmin = NULL,
  include = NULL,
  maxallowed = NULL,
  minallowed = NULL,
  skip_invalid = FALSE
)

go_layout_polar_radialaxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_polar_radialaxis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_layout_polar_radialaxis_title(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
```

```
)

go_layout_polar_radialaxis_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
angularaxis	Named list or `go_layout_polar_angularaxis()` object.
bargap	Sets the gap between bars of adjacent location coordinates. Values are unitless, they represent fractions of the minimum difference in bar positions in the data. Number in the interval [0, 1].
barmode	Determines how bars at the same location coordinate are displayed on the graph. With "stack", the bars are stacked on top of one another With "overlay", the bars are plotted over one another, you might need to reduce "opacity" to see multiple bars. One of `"stack"`, `"overlay"`.
bgcolor	Set the background color of the subplot Color: a CSS color string or R color name.
domain	Named list or `go_layout_polar_domain()` object.
gridshape	Determines if the radial axis grid lines and angular axis line are drawn as "circular" sectors or as "linear" (polygon) sectors. Has an effect only when the angular axis has ‘type‘ "category". Note that ‘radialaxis.angle‘ is snapped to the angle of the closest vertex when ‘gridshape‘ is "circular" (so that radial axis scale is the same as the data scale). One of `"circular"`, `"linear"`.
hole	Sets the fraction of the radius to cut out of the polar subplot. Number in the interval [0, 1].
radialaxis	Named list or `go_layout_polar_radialaxis()` object.
sector	Sets angular span of this polar subplot with two angles (in degrees). Sector are assumed to be spanned in the counterclockwise direction with 0 corresponding to rightmost limit of the polar subplot. Vector or list of 2 element(s).

uirevision
: go_layout_polar(): Controls persistence of user-driven changes in axis attributes, if not overridden in the individual axes. Defaults to 'layout.uirevision'. Any value.

  go_layout_polar_angularaxis(): Controls persistence of user-driven changes in axis 'rotation'. Defaults to 'polar<N>.uirevision'. Any value.

  go_layout_polar_radialaxis(): Controls persistence of user-driven changes in axis 'range', 'autorange', 'angle', and 'title' if in 'editable: true' configuration. Defaults to 'polar<N>.uirevision'. Any value.

skip_invalid
: If TRUE, silently drop invalid properties instead of raising an error.

autotypenumbers
: Using "strict" a numeric string in trace data is not converted to a number. Using "convert types" a numeric string in trace data may be treated as a number during automatic axis 'type' detection. Defaults to layout.autotypenumbers. One of "convert types", "strict".

categoryarray
: Sets the order in which categories on this axis appear. Only has an effect if 'categoryorder' is set to "array". Used with 'categoryorder'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

categoryorder
: Specifies the ordering logic for the case of categorical variables. By default, plotly uses "trace", which specifies the order that is present in the data supplied. Set 'categoryorder' to "category ascending" or "category descending" if order should be determined by the alphanumerical order of the category names. Set 'categoryorder' to "array" to derive the ordering from the attribute 'categoryarray'. If a category is not found in the 'categoryarray' array, the sorting behavior for that attribute will be identical to the "trace" mode. The unspecified categories will follow the categories in 'categoryarray'. Set 'categoryorder' to "total ascending" or "total descending" if order should be determined by the numerical order of the values. Similarly, the order can be determined by the min, max, sum, mean, geometric mean or median of all the values. One of "trace", "category ascending", "category descending", "array", "total ascending", "total descending", "min ascending", "min descending", "max ascending", "max descending", "sum ascending", "sum descending", "mean ascending", "mean descending", "geometric mean ascending", "geometric mean descending", "median ascending", "median descending".

color
: go_layout_polar_angularaxis(), go_layout_polar_radialaxis(): Sets default for all colors associated with this axis all at once: line, font, tick, and grid colors. Grid color is lightened by blending this with the plot background Individual pieces can override this. Color: a CSS color string or R color name.

  go_layout_polar_angularaxis_tickfont(), go_layout_polar_radialaxis_tickfont(), go_layout_polar_radialaxis_title_font(): Color: a CSS color string or R color name.

direction
: Sets the direction corresponding to positive angles. One of "counterclockwise", "clockwise".

dtick
: Sets the step in-between ticks on this axis. Use with 'tick0'. Must be a positive number, or special strings available to "log" and "date" axes. If the axis 'type' is "log", then ticks are set every 10^(n*dtick) where n is the tick number. For example, to set a tick mark at 1, 10, 100, 1000, ... set dtick to 1. To set tick marks

at 1, 100, 10000, ... set dtick to 2. To set tick marks at 1, 5, 25, 125, 625, 3125, ... set dtick to log_10(5), or 0.69897000433. "log" has several special values; "L<f>", where ‘f‘ is a positive number, gives ticks linearly spaced in value (but not position). For example ‘tick0‘ = 0.1, ‘dtick‘ = "L0.5" will put ticks at 0.1, 0.6, 1.1, 1.6 etc. To show powers of 10 plus small digits between, use "D1" (all digits) or "D2" (only 2 and 5). ‘tick0‘ is ignored for "D1" and "D2". If the axis ‘type‘ is "date", then you must convert the time to milliseconds. For example, to set the interval between ticks to one day, set ‘dtick‘ to 86400000.0. "date" also has special values "M<n>" gives ticks spaced by a number of months. ‘n‘ must be a positive integer. To set ticks on the 15th of every third month, set ‘tick0‘ to "2000-01-15" and ‘dtick‘ to "M3". To set ticks every 4 years, set ‘dtick‘ to "M48" Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`gridcolor` Sets the color of the grid lines. Color: a CSS color string or R color name.

`griddash` Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

`gridwidth` Sets the width (in px) of the grid lines. Number in the interval [0, Inf].

`hoverformat` Sets the hover text formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`layer` Sets the layer on which this axis is displayed. If "above traces", this axis is displayed above all the subplot's traces If "below traces", this axis is displayed below all the subplot's traces, but above the grid lines. Useful when used together with scatter-like traces with ‘cliponaxis‘ set to FALSE to show markers and/or text nodes above this axis. One of `"above traces"`, `"below traces"`.

`linecolor` Sets the axis line color. Color: a CSS color string or R color name.

`linewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

`minorloglabels` Determines how minor log labels are displayed. If "small digits", small digits i.e. 2 or 5 are displayed. If "complete", complete digits are displayed. If "none", no labels are displayed. One of `"small digits"`, `"complete"`, `"none"`.

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`period` Set the angular period. Has an effect only when ‘angularaxis.type‘ is "category". Number in the interval [0, Inf].

`rotation` Sets that start position (in degrees) of the angular axis By default, polar subplots with ‘direction‘ set to "counterclockwise" get a ‘rotation‘ of 0 which corresponds to due East (like what mathematicians prefer). In turn, polar with ‘direction‘ set to "clockwise" get a rotation of 90 which corresponds to due North (like on a compass), Angle in degrees.

`separatethousands` If "true", even 4-digit integers are separated Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showgrid` Determines whether or not grid lines are drawn. If TRUE, the grid lines are drawn at every tick mark. Logical.

`showline` Determines whether or not a line bounding this axis is drawn. Logical.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`thetaunit` Sets the format unit of the formatted "theta" values. Has an effect only when ‘angularaxis.type‘ is "linear". One of `"radians"`, `"degrees"`.

`tick0` Sets the placement of the first tick on this axis. Use with ‘dtick‘. If the axis ‘type‘ is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the ‘tick0‘ to 2) except when ‘dtick‘="L<f>" (see ‘dtick‘ for more info). If the axis ‘type‘ is "date", it should be a date string, like date data. If the axis ‘type‘ is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a ‘tickangle‘ of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

tickfont
: go_layout_polar_angularaxis(): Sets the tick font. Named list or go_layout_polar_angularaxis_t object.
go_layout_polar_radialaxis(): Sets the tick font. Named list or go_layout_polar_radialaxis_ti object.

tickformat
: Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops
: go_layout_polar_angularaxis(): List of named lists or go_layout_polar_angularaxis_tickforma objects.
go_layout_polar_radialaxis(): List of named lists or go_layout_polar_radialaxis_tickformats objects.

ticklabelstep
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. 'tick0' determines which labels are shown. Not implemented for axes with 'type' "log" or "multicategory", or when 'tickmode' is "array". Integer in the interval [1, Inf].

ticklen
: Sets the tick length (in px). Number in the interval [0, Inf].

tickmode
: Sets the tick mode for this axis. If "auto", the number of ticks is set via 'nticks'. If "linear", the placement of the ticks is determined by a starting position 'tick0' and a tick step 'dtick' ("linear" is the default value if 'tick0' and 'dtick' are provided). If "array", the placement of the ticks is set via 'tickvals' and the tick text is 'ticktext'. ("array" is the default value if 'tickvals' is provided). One of `"auto"`, `"linear"`, `"array"`.

tickprefix
: Sets a tick label prefix. String.

ticks
: Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

ticksuffix
: Sets a tick label suffix. String.

ticktext
: Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals
: Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth
: Sets the tick width (in px). Number in the interval [0, Inf].

type
: go_layout_polar_angularaxis(): Sets the angular axis type. If "linear", set 'thetaunit' to determine the unit in which axis value are shown. If *category, use 'period' to set the number of integer coordinates around polar axis. One of `"-"`, `"linear"`, `"category"`.

go_layout_polar_radialaxis(): Sets the axis type. By default, plotly attempts to determined the axis type by looking into the data of the traces that referenced the axis in question. One of "-", "linear", "log", "date", "category".

visible — A single toggle to hide the axis while preserving interaction like dragging. Default is true when a cheater plot is present on the axis, otherwise false Logical.

family — HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition — Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow — Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size — Number in the interval [1, Inf].

style — Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase — Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant — Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".

weight — Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

dtickrange — range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

enabled — Determines whether or not this stop is used. If 'false', this stop is ignored even within its 'dtickrange'. Logical.

name — When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

templateitemname — Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

value — string - dtickformat for described zoom level, the same as "tickformat" String.

column — If there is a layout grid, use the domain for this column in the grid for this polar subplot. Integer in the interval [0, Inf].

`row` If there is a layout grid, use the domain for this row in the grid for this polar subplot. Integer in the interval [0, Inf].

`x` Sets the horizontal domain of this polar subplot (in plot fraction). Vector or list of 2 element(s).

`y` Sets the vertical domain of this polar subplot (in plot fraction). Vector or list of 2 element(s).

`angle` Sets the angle (in degrees) from which the radial axis is drawn. Note that by default, radial axis line on the theta=0 line corresponds to a line pointing right (like what mathematicians prefer). Defaults to the first ‘polar.sector‘ angle. Angle in degrees.

`autorange` Determines whether or not the range of this axis is computed in relation to the input data. See ‘rangemode‘ for more info. If ‘range‘ is provided and it has a value for both the lower and upper bound, ‘autorange‘ is set to FALSE. Using "min" applies autorange only to set the minimum. Using "max" applies autorange only to set the maximum. Using "min reversed" applies autorange only to set the minimum on a reversed axis. Using "max reversed" applies autorange only to set the maximum on a reversed axis. Using "reversed" applies autorange on both ends and reverses the axis direction. One of `"reversed"`, `"min reversed"`, `"max reversed"`, `"min"`, `"max"`, `TRUE`, `FALSE`.

`autorangeoptions` Named list or `go_layout_polar_radialaxis_autorangeoptions()` object.

`autotickangles` When ‘tickangle‘ is set to "auto", it will be set to the first angle in this array that is large enough to prevent label overlap. Vector or list of up to 1 element(s).

`calendar` Sets the calendar system to use for ‘range‘ and ‘tick0‘ if this is a date axis. This does not set the calendar for interpreting data on this axis, that’s specified in the trace or via the global ‘layout.calendar‘ One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

`maxallowed` `go_layout_polar_radialaxis()`: Determines the maximum range of this axis. Any value.

`go_layout_polar_radialaxis_autorangeoptions()`: Use this value exactly as autorange maximum. Any value.

`minallowed` `go_layout_polar_radialaxis()`: Determines the minimum range of this axis. Any value.

`go_layout_polar_radialaxis_autorangeoptions()`: Use this value exactly as autorange minimum. Any value.

`range` Sets the range of this axis. If the axis ‘type‘ is "log", then you must take the log of your desired range (e.g. to set the range from 1 to 100, set the range from 0 to 2). If the axis ‘type‘ is "date", it should be date strings, like date data, though Date objects and unix milliseconds will be accepted and converted to strings. If the axis ‘type‘ is "category", it should be numbers, using the scale where each category is assigned a serial number from zero in the order it appears. Leaving either or both elements ‘null‘ impacts the default ‘autorange‘. Vector or list of 2 element(s).

rangemode	If "tozero", the range extends to 0, regardless of the input data If "nonnegative", the range is non-negative, regardless of the input data. If "normal", the range is computed in relation to the extrema of the input data (same behavior as for cartesian axes). One of "tozero", "nonnegative", "normal".
side	Determines on which side of radial axis line the tick and tick labels appear. One of "clockwise", "counterclockwise".
title	String (the title text), named list or go_layout_polar_radialaxis_title() object.
clipmax	Clip autorange maximum if it goes beyond this value. Has no effect when 'autorangeoptions.maxallowed' is provided. Any value.
clipmin	Clip autorange minimum if it goes beyond this value. Has no effect when 'autorangeoptions.minallowed' is provided. Any value.
include	Extends the autorange to include this value or array of values, even when they fall outside the data range. Has no effect on endpoints set by 'autorangeoptions.minallowed' or 'autorangeoptions.maxallowed'. Subject to 'autorangeoptions.clipmin' and 'autorangeoptions.clipmax'. Any value.
font	Sets this axis' title font. Named list or go_layout_polar_radialaxis_title_font() object.
text	Sets the title of this axis. String.

Details

go_layout_polar() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Nested objects documented here:

go_layout_polar_angularaxis() layout.polar.angularaxis

go_layout_polar_angularaxis_tickfont() layout.polar.angularaxis.tickfont: Sets the tick font.

go_layout_polar_angularaxis_tickformatstop() layout.polar.angularaxis.tickformatstop

go_layout_polar_domain() layout.polar.domain

go_layout_polar_radialaxis() layout.polar.radialaxis

go_layout_polar_radialaxis_autorangeoptions() layout.polar.radialaxis.autorangeoptions

go_layout_polar_radialaxis_tickfont() layout.polar.radialaxis.tickfont: Sets the tick font.

go_layout_polar_radialaxis_tickformatstop() layout.polar.radialaxis.tickformatstop

go_layout_polar_radialaxis_title() layout.polar.radialaxis.title

go_layout_polar_radialaxis_title_font() layout.polar.radialaxis.title.font: Sets this axis' title font.

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

---

go_layout_scene *Layout scene objects*

Description

Properties of layout.scene.

Usage

```
go_layout_scene(
  arg = NULL,
  ...,
  annotations = NULL,
  aspectmode = NULL,
  aspectratio = NULL,
  bgcolor = NULL,
  camera = NULL,
  domain = NULL,
  dragmode = NULL,
  hovermode = NULL,
  uirevision = NULL,
  xaxis = NULL,
  yaxis = NULL,
  zaxis = NULL,
  skip_invalid = FALSE
)

go_layout_scene_annotation(
  arg = NULL,
  ...,
  align = NULL,
  arrowcolor = NULL,
  arrowhead = NULL,
  arrowside = NULL,
  arrowsize = NULL,
  arrowwidth = NULL,
  ax = NULL,
  ay = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  borderpad = NULL,
  borderwidth = NULL,
  captureevents = NULL,
```

```
  font = NULL,
  height = NULL,
  hoverlabel = NULL,
  hovertext = NULL,
  name = NULL,
  opacity = NULL,
  showarrow = NULL,
  standoff = NULL,
  startarrowhead = NULL,
  startarrowsize = NULL,
  startstandoff = NULL,
  templateitemname = NULL,
  text = NULL,
  textangle = NULL,
  valign = NULL,
  visible = NULL,
  width = NULL,
  x = NULL,
  xanchor = NULL,
  xshift = NULL,
  y = NULL,
  yanchor = NULL,
  yshift = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_layout_scene_annotation_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_scene_annotation_hoverlabel(
  arg = NULL,
  ...,
  bgcolor = NULL,
  bordercolor = NULL,
  font = NULL,
```

```
  skip_invalid = FALSE
)

go_layout_scene_annotation_hoverlabel_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_scene_aspectratio(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_layout_scene_camera(
  arg = NULL,
  ...,
  center = NULL,
  eye = NULL,
  projection = NULL,
  up = NULL,
  skip_invalid = FALSE
)

go_layout_scene_camera_center(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_layout_scene_camera_eye(
  arg = NULL,
```

```
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_layout_scene_camera_projection(
  arg = NULL,
  ...,
  type = NULL,
  skip_invalid = FALSE
)

go_layout_scene_camera_up(
  arg = NULL,
  ...,
  x = NULL,
  y = NULL,
  z = NULL,
  skip_invalid = FALSE
)

go_layout_scene_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_layout_scene_xaxis(
  arg = NULL,
  ...,
  autorange = NULL,
  autorangeoptions = NULL,
  autotypenumbers = NULL,
  backgroundcolor = NULL,
  calendar = NULL,
  categoryarray = NULL,
  categoryorder = NULL,
  color = NULL,
  dtick = NULL,
  exponentformat = NULL,
  gridcolor = NULL,
  gridwidth = NULL,
```

```
  hoverformat = NULL,
  labelalias = NULL,
  linecolor = NULL,
  linewidth = NULL,
  maxallowed = NULL,
  minallowed = NULL,
  minexponent = NULL,
  mirror = NULL,
  nticks = NULL,
  range = NULL,
  rangemode = NULL,
  separatethousands = NULL,
  showaxeslabels = NULL,
  showbackground = NULL,
  showexponent = NULL,
  showgrid = NULL,
  showline = NULL,
  showspikes = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  spikecolor = NULL,
  spikesides = NULL,
  spikethickness = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  type = NULL,
  visible = NULL,
  zeroline = NULL,
  zerolinecolor = NULL,
  zerolinewidth = NULL,
  skip_invalid = FALSE
)

go_layout_scene_xaxis_autorangeoptions(
```

```
  arg = NULL,
  ...,
  clipmax = NULL,
  clipmin = NULL,
  include = NULL,
  maxallowed = NULL,
  minallowed = NULL,
  skip_invalid = FALSE
)

go_layout_scene_xaxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_scene_xaxis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_layout_scene_xaxis_title(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_layout_scene_xaxis_title_font(
  arg = NULL,
  ...,
  color = NULL,
```

```
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_scene_yaxis(
  arg = NULL,
  ...,
  autorange = NULL,
  autorangeoptions = NULL,
  autotypenumbers = NULL,
  backgroundcolor = NULL,
  calendar = NULL,
  categoryarray = NULL,
  categoryorder = NULL,
  color = NULL,
  dtick = NULL,
  exponentformat = NULL,
  gridcolor = NULL,
  gridwidth = NULL,
  hoverformat = NULL,
  labelalias = NULL,
  linecolor = NULL,
  linewidth = NULL,
  maxallowed = NULL,
  minallowed = NULL,
  minexponent = NULL,
  mirror = NULL,
  nticks = NULL,
  range = NULL,
  rangemode = NULL,
  separatethousands = NULL,
  showaxeslabels = NULL,
  showbackground = NULL,
  showexponent = NULL,
  showgrid = NULL,
  showline = NULL,
  showspikes = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  spikecolor = NULL,
```

```
  spikesides = NULL,
  spikethickness = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  type = NULL,
  visible = NULL,
  zeroline = NULL,
  zerolinecolor = NULL,
  zerolinewidth = NULL,
  skip_invalid = FALSE
)

go_layout_scene_yaxis_autorangeoptions(
  arg = NULL,
  ...,
  clipmax = NULL,
  clipmin = NULL,
  include = NULL,
  maxallowed = NULL,
  minallowed = NULL,
  skip_invalid = FALSE
)

go_layout_scene_yaxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
```

```
  skip_invalid = FALSE
)

go_layout_scene_yaxis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_layout_scene_yaxis_title(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_layout_scene_yaxis_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_scene_zaxis(
  arg = NULL,
  ...,
  autorange = NULL,
  autorangeoptions = NULL,
  autotypenumbers = NULL,
  backgroundcolor = NULL,
  calendar = NULL,
  categoryarray = NULL,
  categoryorder = NULL,
  color = NULL,
```

```
    dtick = NULL,
    exponentformat = NULL,
    gridcolor = NULL,
    gridwidth = NULL,
    hoverformat = NULL,
    labelalias = NULL,
    linecolor = NULL,
    linewidth = NULL,
    maxallowed = NULL,
    minallowed = NULL,
    minexponent = NULL,
    mirror = NULL,
    nticks = NULL,
    range = NULL,
    rangemode = NULL,
    separatethousands = NULL,
    showaxeslabels = NULL,
    showbackground = NULL,
    showexponent = NULL,
    showgrid = NULL,
    showline = NULL,
    showspikes = NULL,
    showticklabels = NULL,
    showtickprefix = NULL,
    showticksuffix = NULL,
    spikecolor = NULL,
    spikesides = NULL,
    spikethickness = NULL,
    tick0 = NULL,
    tickangle = NULL,
    tickcolor = NULL,
    tickfont = NULL,
    tickformat = NULL,
    tickformatstops = NULL,
    ticklen = NULL,
    tickmode = NULL,
    tickprefix = NULL,
    ticks = NULL,
    ticksuffix = NULL,
    ticktext = NULL,
    tickvals = NULL,
    tickwidth = NULL,
    title = NULL,
    type = NULL,
    visible = NULL,
    zeroline = NULL,
    zerolinecolor = NULL,
    zerolinewidth = NULL,
```

```
  skip_invalid = FALSE
)

go_layout_scene_zaxis_autorangeoptions(
  arg = NULL,
  ...,
  clipmax = NULL,
  clipmin = NULL,
  include = NULL,
  maxallowed = NULL,
  minallowed = NULL,
  skip_invalid = FALSE
)

go_layout_scene_zaxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_scene_zaxis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_layout_scene_zaxis_title(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)
```

```
go_layout_scene_zaxis_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
annotations	List of named lists or `go_layout_scene_annotation()` objects.
aspectmode	If "cube", this scene's axes are drawn as a cube, regardless of the axes' ranges. If "data", this scene's axes are drawn in proportion with the axes' ranges. If "manual", this scene's axes are drawn in proportion with the input of "aspectratio" (the default behavior if "aspectratio" is provided). If "auto", this scene's axes are drawn using the results of "data" except when one axis is more than four times the size of the two others, where in that case the results of "cube" are used. One of `"auto"`, `"cube"`, `"data"`, `"manual"`.
aspectratio	Sets this scene's axis aspectratio. Named list or `go_layout_scene_aspectratio()` object.
bgcolor	`go_layout_scene()`: Color: a CSS color string or R color name. `go_layout_scene_annotation()`: Sets the background color of the annotation. Color: a CSS color string or R color name. `go_layout_scene_annotation_hoverlabel()`: Sets the background color of the hover label. By default uses the annotation's 'bgcolor' made opaque, or white if it was transparent. Color: a CSS color string or R color name.
camera	Named list or `go_layout_scene_camera()` object.
domain	Named list or `go_layout_scene_domain()` object.
dragmode	Determines the mode of drag interactions for this scene. One of `"orbit"`, `"turntable"`, `"zoom"`, `"pan"`, `FALSE`.
hovermode	Determines the mode of hover interactions for this scene. One of `"closest"`, `FALSE`.
uirevision	Controls persistence of user-driven changes in camera attributes. Defaults to 'layout.uirevision'. Any value.

xaxis	Named list or go_layout_scene_xaxis() object.
yaxis	Named list or go_layout_scene_yaxis() object.
zaxis	Named list or go_layout_scene_zaxis() object.
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
align	Sets the horizontal alignment of the 'text' within the box. Has an effect only if 'text' spans two or more lines (i.e. 'text' contains one or more HTML tags) or if an explicit width is set to override the text width. One of "left", "center", "right".
arrowcolor	Sets the color of the annotation arrow. Color: a CSS color string or R color name.
arrowhead	Sets the end annotation arrow head style. Integer in the interval [0, 8].
arrowside	Sets the annotation arrow head position. Any combination of "end", "start" joined with "+", or one of "none".
arrowsize	Sets the size of the end annotation arrow head, relative to 'arrowwidth'. A value of 1 (default) gives a head about 3x as wide as the line. Number in the interval [0.3, Inf].
arrowwidth	Sets the width (in px) of annotation arrow line. Number in the interval [0.1, Inf].
ax	Sets the x component of the arrow tail about the arrow head (in pixels). Number.
ay	Sets the y component of the arrow tail about the arrow head (in pixels). Number.
bordercolor	go_layout_scene_annotation(): Sets the color of the border enclosing the annotation 'text'. Color: a CSS color string or R color name. go_layout_scene_annotation_hoverlabel(): Sets the border color of the hover label. By default uses either dark grey or white, for maximum contrast with 'hoverlabel.bgcolor'. Color: a CSS color string or R color name.
borderpad	Sets the padding (in px) between the 'text' and the enclosing border. Number in the interval [0, Inf].
borderwidth	Sets the width (in px) of the border enclosing the annotation 'text'. Number in the interval [0, Inf].
captureevents	Determines whether the annotation text box captures mouse move and click events, or allows those events to pass through to data points in the plot that may be behind the annotation. By default 'captureevents' is FALSE unless 'hovertext' is provided. If you use the event 'plotly_clickannotation' without 'hovertext' you must explicitly enable 'captureevents'. Logical.
font	go_layout_scene_annotation(): Sets the annotation text font. Named list or go_layout_scene_annotation_font() object. go_layout_scene_annotation_hoverlabel(): Sets the hover label text font. By default uses the global hover font and size, with color from 'hoverlabel.bordercolor'. Named list or go_layout_scene_annotation_hoverlabel_font() object. go_layout_scene_xaxis_title(): Sets this axis' title font. Named list or go_layout_scene_xaxis_title_font() object. go_layout_scene_yaxis_title(): Sets this axis' title font. Named list or go_layout_scene_yaxis_title_font() object. go_layout_scene_zaxis_title(): Sets this axis' title font. Named list or go_layout_scene_zaxis_title_font() object.

height Sets an explicit height for the text box. null (default) lets the text set the box height. Taller text will be clipped. Number in the interval [1, Inf].

hoverlabel Named list or go_layout_scene_annotation_hoverlabel() object.

hovertext Sets text to appear when hovering over this annotation. If omitted or blank, no hover label will appear. String.

name When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with 'templateitemname' matching this 'name' alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). Has no effect outside of a template. String.

opacity Sets the opacity of the annotation (text + arrow). Number in the interval [0, 1].

showarrow Determines whether or not the annotation is drawn with an arrow. If TRUE, 'text' is placed near the arrow's tail. If FALSE, 'text' lines up with the 'x' and 'y' provided. Logical.

standoff Sets a distance, in pixels, to move the end arrowhead away from the position it is pointing at, for example to point at the edge of a marker independent of zoom. Note that this shortens the arrow from the 'ax' / 'ay' vector, in contrast to 'xshift' / 'yshift' which moves everything by this amount. Number in the interval [0, Inf].

startarrowhead Sets the start annotation arrow head style. Integer in the interval [0, 8].

startarrowsize Sets the size of the start annotation arrow head, relative to 'arrowwidth'. A value of 1 (default) gives a head about 3x as wide as the line. Number in the interval [0.3, Inf].

startstandoff Sets a distance, in pixels, to move the start arrowhead away from the position it is pointing at, for example to point at the edge of a marker independent of zoom. Note that this shortens the arrow from the 'ax' / 'ay' vector, in contrast to 'xshift' / 'yshift' which moves everything by this amount. Number in the interval [0, Inf].

templateitemname Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with 'templateitemname' matching its 'name', alongside your modifications (including 'visible: false' or 'enabled: false' to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with 'visible: true'. String.

text go_layout_scene_annotation(): Sets the text associated with this annotation. Plotly uses a subset of HTML tags to do things like newline ('<br>'), bold (''), italics (''), hyperlinks (''). Tags '', '<sup>', '<sub>', '<s>', '<u>', and '' are also supported. String.

go_layout_scene_xaxis_title(), go_layout_scene_yaxis_title(), go_layout_scene_zaxis_ti
Sets the title of this axis. String.

textangle Sets the angle at which the 'text' is drawn with respect to the horizontal. Angle in degrees.

valign Sets the vertical alignment of the 'text' within the box. Has an effect only if an explicit height is set to override the text height. One of "top", "middle", "bottom".

visible
: go_layout_scene_annotation(): Determines whether or not this annotation is visible. Logical.

  go_layout_scene_xaxis(), go_layout_scene_yaxis(), go_layout_scene_zaxis(): A single toggle to hide the axis while preserving interaction like dragging. Default is true when a cheater plot is present on the axis, otherwise false Logical.

width
: Sets an explicit width for the text box. null (default) lets the text set the box width. Wider text will be clipped. There is no automatic wrapping; use <br> to start a new line. Number in the interval [1, Inf].

x
: go_layout_scene_annotation(): Sets the annotation's x position. Any value.

  go_layout_scene_aspectratio(): Number in the interval [0, Inf].

  go_layout_scene_camera_center(), go_layout_scene_camera_eye(), go_layout_scene_camera_ Number.

  go_layout_scene_domain(): Sets the horizontal domain of this scene subplot (in plot fraction). Vector or list of 2 element(s).

xanchor
: Sets the text box's horizontal position anchor This anchor binds the 'x' position to the "left", "center" or "right" of the annotation. For example, if 'x' is set to 1, 'xref' to "paper" and 'xanchor' to "right" then the right-most portion of the annotation lines up with the right-most edge of the plotting area. If "auto", the anchor is equivalent to "center" for data-referenced annotations or if there is an arrow, whereas for paper-referenced with no arrow, the anchor picked corresponds to the closest side. One of "auto", "left", "center", "right".

xshift
: Shifts the position of the whole annotation and arrow to the right (positive) or left (negative) by this many pixels. Number.

y
: go_layout_scene_annotation(): Sets the annotation's y position. Any value.

  go_layout_scene_aspectratio(): Number in the interval [0, Inf].

  go_layout_scene_camera_center(), go_layout_scene_camera_eye(), go_layout_scene_camera_ Number.

  go_layout_scene_domain(): Sets the vertical domain of this scene subplot (in plot fraction). Vector or list of 2 element(s).

yanchor
: Sets the text box's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the annotation. For example, if 'y' is set to 1, 'yref' to "paper" and 'yanchor' to "top" then the top-most portion of the annotation lines up with the top-most edge of the plotting area. If "auto", the anchor is equivalent to "middle" for data-referenced annotations or if there is an arrow, whereas for paper-referenced with no arrow, the anchor picked corresponds to the closest side. One of "auto", "top", "middle", "bottom".

yshift
: Shifts the position of the whole annotation and arrow up (positive) or down (negative) by this many pixels. Number.

z
: go_layout_scene_annotation(): Sets the annotation's z position. Any value.

  go_layout_scene_aspectratio(): Number in the interval [0, Inf].

  go_layout_scene_camera_center(), go_layout_scene_camera_eye(), go_layout_scene_camera_ Number.

color
: go_layout_scene_annotation_font(), go_layout_scene_annotation_hoverlabel_font(), go_layout_scene_xaxis_tickfont(), go_layout_scene_xaxis_title_font(),

	`go_layout_scene_yaxis_tickfont()`, `go_layout_scene_yaxis_title_font()`, `go_layout_scene_zaxis_tickfont()`, `go_layout_scene_zaxis_title_font()`: Color: a CSS color string or R color name.
	`go_layout_scene_xaxis()`, `go_layout_scene_yaxis()`, `go_layout_scene_zaxis()`: Sets default for all colors associated with this axis all at once: line, font, tick, and grid colors. Grid color is lightened by blending this with the plot background Individual pieces can override this. Color: a CSS color string or R color name.
`family`	HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.
`lineposition`	Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.
`shadow`	Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.
`size`	Number in the interval [1, Inf].
`style`	Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.
`textcase`	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.
`variant`	Sets the variant of the font. One of `"normal"`, `"small-caps"`, `"all-small-caps"`, `"all-petite-caps"`, `"petite-caps"`, `"unicase"`.
`weight`	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
`center`	Sets the (x,y,z) components of the 'center' camera vector This vector determines the translation (x,y,z) space about the center of this scene. By default, there is no such translation. Named list or `go_layout_scene_camera_center()` object.
`eye`	Sets the (x,y,z) components of the 'eye' camera vector. This vector determines the view point about the origin of this scene. Named list or `go_layout_scene_camera_eye()` object.
`projection`	Named list or `go_layout_scene_camera_projection()` object.
`up`	Sets the (x,y,z) components of the 'up' camera vector. This vector determines the up direction of this scene with respect to the page. The default is "{x: 0, y: 0, z: 1}" which means that the z axis points up. Named list or `go_layout_scene_camera_up()` object.
`type`	`go_layout_scene_camera_projection()`: Sets the projection type. The projection type could be either "perspective" or "orthographic". The default is "perspective". One of `"perspective"`, `"orthographic"`.
	`go_layout_scene_xaxis()`, `go_layout_scene_yaxis()`, `go_layout_scene_zaxis()`: Sets the axis type. By default, plotly attempts to determined the axis type by looking into the data of the traces that referenced the axis in question. One of `"-"`, `"linear"`, `"log"`, `"date"`, `"category"`.

column
: If there is a layout grid, use the domain for this column in the grid for this scene subplot. Integer in the interval [0, Inf].

row
: If there is a layout grid, use the domain for this row in the grid for this scene subplot. Integer in the interval [0, Inf].

autorange
: Determines whether or not the range of this axis is computed in relation to the input data. See ‘rangemode‘ for more info. If ‘range‘ is provided and it has a value for both the lower and upper bound, ‘autorange‘ is set to FALSE. Using "min" applies autorange only to set the minimum. Using "max" applies autorange only to set the maximum. Using "min reversed" applies autorange only to set the minimum on a reversed axis. Using "max reversed" applies autorange only to set the maximum on a reversed axis. Using "reversed" applies autorange on both ends and reverses the axis direction. One of `"reversed"`, `"min reversed"`, `"max reversed"`, `"min"`, `"max"`, `TRUE`, `FALSE`.

autorangeoptions
: `go_layout_scene_xaxis()`: Named list or `go_layout_scene_xaxis_autorangeoptions()` object.

  `go_layout_scene_yaxis()`: Named list or `go_layout_scene_yaxis_autorangeoptions()` object.

  `go_layout_scene_zaxis()`: Named list or `go_layout_scene_zaxis_autorangeoptions()` object.

autotypenumbers
: Using "strict" a numeric string in trace data is not converted to a number. Using "convert types" a numeric string in trace data may be treated as a number during automatic axis ‘type‘ detection. Defaults to layout.autotypenumbers. One of `"convert types"`, `"strict"`.

backgroundcolor
: Sets the background color of this axis’ wall. Color: a CSS color string or R color name.

calendar
: Sets the calendar system to use for ‘range‘ and ‘tick0‘ if this is a date axis. This does not set the calendar for interpreting data on this axis, that’s specified in the trace or via the global ‘layout.calendar‘ One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

categoryarray
: Sets the order in which categories on this axis appear. Only has an effect if ‘categoryorder‘ is set to "array". Used with ‘categoryorder‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

categoryorder
: Specifies the ordering logic for the case of categorical variables. By default, plotly uses "trace", which specifies the order that is present in the data supplied. Set ‘categoryorder‘ to "category ascending" or "category descending" if order should be determined by the alphanumerical order of the category names. Set ‘categoryorder‘ to "array" to derive the ordering from the attribute ‘categoryarray‘. If a category is not found in the ‘categoryarray‘ array, the sorting behavior for that attribute will be identical to the "trace" mode. The unspecified categories will follow the categories in ‘categoryarray‘. Set ‘categoryorder‘ to "total ascending" or "total descending" if order should be deter-

mined by the numerical order of the values. Similarly, the order can be determined by the min, max, sum, mean, geometric mean or median of all the values. One of `"trace"`, `"category ascending"`, `"category descending"`, `"array"`, `"total ascending"`, `"total descending"`, `"min ascending"`, `"min descending"`, `"max ascending"`, `"max descending"`, `"sum ascending"`, `"sum descending"`, `"mean ascending"`, `"mean descending"`, `"geometric mean ascending"`, `"geometric mean descending"`, `"median ascending"`, `"median descending"`.

`dtick` Sets the step in-between ticks on this axis. Use with ‘tick0‘. Must be a positive number, or special strings available to "log" and "date" axes. If the axis ‘type‘ is "log", then ticks are set every 10^(n*dtick) where n is the tick number. For example, to set a tick mark at 1, 10, 100, 1000, ... set dtick to 1. To set tick marks at 1, 100, 10000, ... set dtick to 2. To set tick marks at 1, 5, 25, 125, 625, 3125, ... set dtick to log_10(5), or 0.69897000433. "log" has several special values; "L<f>", where ‘f‘ is a positive number, gives ticks linearly spaced in value (but not position). For example ‘tick0‘ = 0.1, ‘dtick‘ = "L0.5" will put ticks at 0.1, 0.6, 1.1, 1.6 etc. To show powers of 10 plus small digits between, use "D1" (all digits) or "D2" (only 2 and 5). ‘tick0‘ is ignored for "D1" and "D2". If the axis ‘type‘ is "date", then you must convert the time to milliseconds. For example, to set the interval between ticks to one day, set ‘dtick‘ to 86400000.0. "date" also has special values "M<n>" gives ticks spaced by a number of months. ‘n‘ must be a positive integer. To set ticks on the 15th of every third month, set ‘tick0‘ to "2000-01-15" and ‘dtick‘ to "M3". To set ticks every 4 years, set ‘dtick‘ to "M48" Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`gridcolor` Sets the color of the grid lines. Color: a CSS color string or R color name.

`gridwidth` Sets the width (in px) of the grid lines. Number in the interval [0, Inf].

`hoverformat` Sets the hover text formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`labelalias` Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

linecolor Sets the axis line color. Color: a CSS color string or R color name.

linewidth Sets the width (in px) of the axis line. Number in the interval [0, Inf].

maxallowed `go_layout_scene_xaxis()`, `go_layout_scene_yaxis()`, `go_layout_scene_zaxis()`: Determines the maximum range of this axis. Any value.
`go_layout_scene_xaxis_autorangeoptions()`, `go_layout_scene_yaxis_autorangeoptions()`, `go_layout_scene_zaxis_autorangeoptions()`: Use this value exactly as autorange maximum. Any value.

minallowed `go_layout_scene_xaxis()`, `go_layout_scene_yaxis()`, `go_layout_scene_zaxis()`: Determines the minimum range of this axis. Any value.
`go_layout_scene_xaxis_autorangeoptions()`, `go_layout_scene_yaxis_autorangeoptions()`, `go_layout_scene_zaxis_autorangeoptions()`: Use this value exactly as autorange minimum. Any value.

minexponent Hide SI prefix for 10^n if |n| is below this number. This only has an effect when ‘tickformat‘ is "SI" or "B". Number in the interval [0, Inf].

mirror Determines if the axis lines or/and ticks are mirrored to the opposite side of the plotting area. If TRUE, the axis lines are mirrored. If "ticks", the axis lines and ticks are mirrored. If FALSE, mirroring is disable. If "all", axis lines are mirrored on all shared-axes subplots. If "allticks", axis lines and ticks are mirrored on all shared-axes subplots. One of `"ticks"`, `"all"`, `"allticks"`, `TRUE`, `FALSE`.

nticks Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

range Sets the range of this axis. If the axis ‘type‘ is "log", then you must take the log of your desired range (e.g. to set the range from 1 to 100, set the range from 0 to 2). If the axis ‘type‘ is "date", it should be date strings, like date data, though Date objects and unix milliseconds will be accepted and converted to strings. If the axis ‘type‘ is "category", it should be numbers, using the scale where each category is assigned a serial number from zero in the order it appears. Leaving either or both elements ‘null‘ impacts the default ‘autorange‘. Vector or list of 2 element(s).

rangemode If "normal", the range is computed in relation to the extrema of the input data. If "tozero", the range extends to 0, regardless of the input data If "nonnegative", the range is non-negative, regardless of the input data. Applies only to linear axes. One of `"normal"`, `"tozero"`, `"nonnegative"`.

separatethousands
If "true", even 4-digit integers are separated Logical.

showaxeslabels Sets whether or not this axis is labeled Logical.

showbackground Sets whether or not this axis' wall has a background color. Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

showgrid Determines whether or not grid lines are drawn. If TRUE, the grid lines are drawn at every tick mark. Logical.

`showline` Determines whether or not a line bounding this axis is drawn. Logical.

`showspikes` Sets whether or not spikes starting from data points to this axis' wall are shown on hover. Logical.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`spikecolor` Sets the color of the spikes. Color: a CSS color string or R color name.

`spikesides` Sets whether or not spikes extending from the projection data points to this axis' wall boundaries are shown on hover. Logical.

`spikethickness` Sets the thickness (in px) of the spikes. Number in the interval [0, Inf].

`tick0` Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

`tickangle` Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

`tickcolor` Sets the tick color. Color: a CSS color string or R color name.

`tickfont`

`go_layout_scene_xaxis()`: Sets the tick font. Named list or `go_layout_scene_xaxis_tickfont()` object.

`go_layout_scene_yaxis()`: Sets the tick font. Named list or `go_layout_scene_yaxis_tickfont()` object.

`go_layout_scene_zaxis()`: Sets the tick font. Named list or `go_layout_scene_zaxis_tickfont()` object.

`tickformat` Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`tickformatstops`

`go_layout_scene_xaxis()`: List of named lists or `go_layout_scene_xaxis_tickformatstop()` objects.

`go_layout_scene_yaxis()`: List of named lists or `go_layout_scene_yaxis_tickformatstop()` objects.

`go_layout_scene_zaxis()`: List of named lists or `go_layout_scene_zaxis_tickformatstop()` objects.

`ticklen` Sets the tick length (in px). Number in the interval [0, Inf].

tickmode	Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.
tickprefix	Sets a tick label prefix. String.
ticks	Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.
ticksuffix	Sets a tick label suffix. String.
ticktext	Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
tickvals	Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
tickwidth	Sets the tick width (in px). Number in the interval [0, Inf].
title	`go_layout_scene_xaxis()`: String (the title text), named list or `go_layout_scene_xaxis_title()` object. `go_layout_scene_yaxis()`: String (the title text), named list or `go_layout_scene_yaxis_title()` object. `go_layout_scene_zaxis()`: String (the title text), named list or `go_layout_scene_zaxis_title()` object.
zeroline	Determines whether or not a line is drawn at along the 0 value of this axis. If TRUE, the zero line is drawn on top of the grid lines. Logical.
zerolinecolor	Sets the line color of the zero line. Color: a CSS color string or R color name.
zerolinewidth	Sets the width (in px) of the zero line. Number.
clipmax	Clip autorange maximum if it goes beyond this value. Has no effect when ‘autorangeoptions.maxallowed‘ is provided. Any value.
clipmin	Clip autorange minimum if it goes beyond this value. Has no effect when ‘autorangeoptions.minallowed‘ is provided. Any value.
include	Extends the autorange to include this value or array of values, even when they fall outside the data range. Has no effect on endpoints set by ‘autorangeoptions.minallowed‘ or ‘autorangeoptions.maxallowed‘. Subject to ‘autorangeoptions.clipmin‘ and ‘autorangeoptions.clipmax‘. Any value.
dtickrange	range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).
enabled	Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.
value	string - dtickformat for described zoom level, the same as "tickformat" String.

Details

`go_layout_scene()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`).

Nested objects documented here:

`go_layout_scene_annotation()` `layout.scene.annotation`

`go_layout_scene_annotation_font()` `layout.scene.annotation.font`: Sets the annotation text font.

`go_layout_scene_annotation_hoverlabel()` `layout.scene.annotation.hoverlabel`

`go_layout_scene_annotation_hoverlabel_font()` `layout.scene.annotation.hoverlabel.font`: Sets the hover label text font. By default uses the global hover font and size, with color from 'hoverlabel.bordercolor'.

`go_layout_scene_aspectratio()` `layout.scene.aspectratio`: Sets this scene's axis aspectratio.

`go_layout_scene_camera()` `layout.scene.camera`

`go_layout_scene_camera_center()` `layout.scene.camera.center`: Sets the (x,y,z) components of the 'center' camera vector This vector determines the translation (x,y,z) space about the center of this scene. By default, there is no such translation.

`go_layout_scene_camera_eye()` `layout.scene.camera.eye`: Sets the (x,y,z) components of the 'eye' camera vector. This vector determines the view point about the origin of this scene.

`go_layout_scene_camera_projection()` `layout.scene.camera.projection`

`go_layout_scene_camera_up()` `layout.scene.camera.up`: Sets the (x,y,z) components of the 'up' camera vector. This vector determines the up direction of this scene with respect to the page. The default is "{x: 0, y: 0, z: 1}" which means that the z axis points up.

`go_layout_scene_domain()` `layout.scene.domain`

`go_layout_scene_xaxis()` `layout.scene.xaxis`

`go_layout_scene_xaxis_autorangeoptions()` `layout.scene.xaxis.autorangeoptions`

`go_layout_scene_xaxis_tickfont()` `layout.scene.xaxis.tickfont`: Sets the tick font.

`go_layout_scene_xaxis_tickformatstop()` `layout.scene.xaxis.tickformatstop`

`go_layout_scene_xaxis_title()` `layout.scene.xaxis.title`

`go_layout_scene_xaxis_title_font()` `layout.scene.xaxis.title.font`: Sets this axis' title font.

`go_layout_scene_yaxis()` `layout.scene.yaxis`

`go_layout_scene_yaxis_autorangeoptions()` `layout.scene.yaxis.autorangeoptions`

`go_layout_scene_yaxis_tickfont()` `layout.scene.yaxis.tickfont`: Sets the tick font.

`go_layout_scene_yaxis_tickformatstop()` `layout.scene.yaxis.tickformatstop`

`go_layout_scene_yaxis_title()` `layout.scene.yaxis.title`

`go_layout_scene_yaxis_title_font()` `layout.scene.yaxis.title.font`: Sets this axis' title font.

`go_layout_scene_zaxis()` `layout.scene.zaxis`

`go_layout_scene_zaxis_autorangeoptions()` `layout.scene.zaxis.autorangeoptions`

`go_layout_scene_zaxis_tickfont()` `layout.scene.zaxis.tickfont`: Sets the tick font.

`go_layout_scene_zaxis_tickformatstop()` `layout.scene.zaxis.tickformatstop`

`go_layout_scene_zaxis_title()` `layout.scene.zaxis.title`

`go_layout_scene_zaxis_title_font()` `layout.scene.zaxis.title.font`: Sets this axis' title font.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

`go_layout_slider` *Layout slider objects*

Description

Properties of `layout.slider`.

Usage

```
go_layout_slider(
  arg = NULL,
  ...,
  active = NULL,
  activebgcolor = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  currentvalue = NULL,
  font = NULL,
  len = NULL,
  lenmode = NULL,
  minorticklen = NULL,
  name = NULL,
  pad = NULL,
  steps = NULL,
  templateitemname = NULL,
  tickcolor = NULL,
  ticklen = NULL,
  tickwidth = NULL,
  transition = NULL,
```

```
  visible = NULL,
  x = NULL,
  xanchor = NULL,
  y = NULL,
  yanchor = NULL,
  skip_invalid = FALSE
)

go_layout_slider_currentvalue(
  arg = NULL,
  ...,
  font = NULL,
  offset = NULL,
  prefix = NULL,
  suffix = NULL,
  visible = NULL,
  xanchor = NULL,
  skip_invalid = FALSE
)

go_layout_slider_currentvalue_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_slider_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
```

```
)

go_layout_slider_pad(
  arg = NULL,
  ...,
  b = NULL,
  l = NULL,
  r = NULL,
  t = NULL,
  skip_invalid = FALSE
)

go_layout_slider_step(
  arg = NULL,
  ...,
  args = NULL,
  execute = NULL,
  label = NULL,
  method = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_layout_slider_transition(
  arg = NULL,
  ...,
  duration = NULL,
  easing = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
active	Determines which button (by index starting from 0) is considered active. Number in the interval [0, Inf].
activebgcolor	Sets the background color of the slider grip while dragging. Color: a CSS color string or R color name.
bgcolor	Sets the background color of the slider. Color: a CSS color string or R color name.

`bordercolor` Sets the color of the border enclosing the slider. Color: a CSS color string or R color name.

`borderwidth` Sets the width (in px) of the border enclosing the slider. Number in the interval [0, Inf].

`currentvalue` Named list or `go_layout_slider_currentvalue()` object.

`font` `go_layout_slider()`: Sets the font of the slider step labels. Named list or `go_layout_slider_font()` object.

`go_layout_slider_currentvalue()`: Sets the font of the current value label text. Named list or `go_layout_slider_currentvalue_font()` object.

`len` Sets the length of the slider This measure excludes the padding of both ends. That is, the slider's length is this length minus the padding on both ends. Number in the interval [0, Inf].

`lenmode` Determines whether this slider length is set in units of plot "fraction" or in *pixels. Use ‘len‘ to set the value. One of `"fraction"`, `"pixels"`.

`minorticklen` Sets the length in pixels of minor step tick marks Number in the interval [0, Inf].

`name` When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`pad` Set the padding of the slider component along each side. Named list or `go_layout_slider_pad()` object.

`steps` List of named lists or `go_layout_slider_step()` objects.

`templateitemname` Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`tickcolor` Sets the color of the border enclosing the slider. Color: a CSS color string or R color name.

`ticklen` Sets the length in pixels of step tick marks Number in the interval [0, Inf].

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`transition` Named list or `go_layout_slider_transition()` object.

`visible` `go_layout_slider()`: Determines whether or not the slider is visible. Logical.

`go_layout_slider_currentvalue()`: Shows the currently-selected value above the slider. Logical.

`go_layout_slider_step()`: Determines whether or not this step is included in the slider. Logical.

`x` Sets the x position (in normalized coordinates) of the slider. Number in the interval [-2, 3].

xanchor	go_layout_slider(): Sets the slider's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the range selector. One of "auto", "left", "center", "right". go_layout_slider_currentvalue(): The alignment of the value readout relative to the length of the slider. One of "left", "center", "right".
y	Sets the y position (in normalized coordinates) of the slider. Number in the interval [-2, 3].
yanchor	Sets the slider's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the range selector. One of "auto", "top", "middle", "bottom".
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.
offset	The amount of space, in pixels, between the current value label and the slider. Number.
prefix	When currentvalue.visible is true, this sets the prefix of the label. String.
suffix	When currentvalue.visible is true, this sets the suffix of the label. String.
color	Color: a CSS color string or R color name.
family	HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.
lineposition	Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".
shadow	Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.
size	Number in the interval [1, Inf].
style	Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".
textcase	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".
variant	Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
weight	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
b	The amount of padding (in px) along the bottom of the component. Number.
l	The amount of padding (in px) on the left side of the component. Number.
r	The amount of padding (in px) on the right side of the component. Number.
t	The amount of padding (in px) along the top of the component. Number.
args	Sets the arguments values to be passed to the Plotly method set in 'method' on slide. Vector or list of up to 3 element(s).

execute	When true, the API method is executed. When false, all other behaviors are the same and command execution is skipped. This may be useful when hooking into, for example, the 'plotly_sliderchange' method and executing the API command manually without losing the benefit of the slider automatically binding to the state of the plot through the specification of 'method' and 'args'. Logical.
label	Sets the text label to appear on the slider String.
method	Sets the Plotly method to be called when the slider value is changed. If the 'skip' method is used, the API slider will function as normal but will perform no API calls and will not bind automatically to state updates. This may be used to create a component interface and attach to slider events manually via JavaScript. One of "restyle", "relayout", "animate", "update", "skip".
value	Sets the value of the slider step, used to refer to the step programatically. Defaults to the slider label if not provided. String.
duration	Sets the duration of the slider transition Number in the interval [0, Inf].
easing	Sets the easing function of the slider transition One of "linear", "quad", "cubic", "sin", "exp", "circle", "elastic", "back", "bounce", "linear-in", "quad-in", "cubic-in", "sin-in", "exp-in", "circle-in", "elastic-in", "back-in", "bounce-in", "linear-out", "quad-out", "cubic-out", "sin-out", "exp-out", "circle-out", "elastic-out", "back-out", "bounce-out", "linear-in-out", "quad-in-out", "cubic-in-out", "sin-in-out", "exp-in-out", "circle-in-out", "elastic-in-out", "back-in-out", "bounce-in-out".

Details

go_layout_slider() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Nested objects documented here:

go_layout_slider_currentvalue() layout.slider.currentvalue

go_layout_slider_currentvalue_font() layout.slider.currentvalue.font: Sets the font of the current value label text.

go_layout_slider_font() layout.slider.font: Sets the font of the slider step labels.

go_layout_slider_pad() layout.slider.pad: Set the padding of the slider component along each side.

go_layout_slider_step() layout.slider.step

go_layout_slider_transition() layout.slider.transition

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

go_layout_smith *Layout smith objects*

Description

Properties of `layout.smith`.

Usage

```
go_layout_smith(
  arg = NULL,
  ...,
  bgcolor = NULL,
  domain = NULL,
  imaginaryaxis = NULL,
  realaxis = NULL,
  skip_invalid = FALSE
)

go_layout_smith_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)

go_layout_smith_imaginaryaxis(
  arg = NULL,
  ...,
  color = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
  hoverformat = NULL,
  labelalias = NULL,
  layer = NULL,
  linecolor = NULL,
  linewidth = NULL,
  showgrid = NULL,
  showline = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  tickcolor = NULL,
```

```
  tickfont = NULL,
  tickformat = NULL,
  ticklen = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_layout_smith_imaginaryaxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_smith_realaxis(
  arg = NULL,
  ...,
  color = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
  hoverformat = NULL,
  labelalias = NULL,
  layer = NULL,
  linecolor = NULL,
  linewidth = NULL,
  showgrid = NULL,
  showline = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  side = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
```

```
  tickformat = NULL,
  ticklen = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_layout_smith_realaxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
bgcolor	Set the background color of the subplot Color: a CSS color string or R color name.
domain	Named list or `go_layout_smith_domain()` object.
imaginaryaxis	Named list or `go_layout_smith_imaginaryaxis()` object.
realaxis	Named list or `go_layout_smith_realaxis()` object.
skip_invalid	If `TRUE`, silently drop invalid properties instead of raising an error.
column	If there is a layout grid, use the domain for this column in the grid for this smith subplot. Integer in the interval [0, Inf].
row	If there is a layout grid, use the domain for this row in the grid for this smith subplot. Integer in the interval [0, Inf].
x	Sets the horizontal domain of this smith subplot (in plot fraction). Vector or list of 2 element(s).

y	Sets the vertical domain of this smith subplot (in plot fraction). Vector or list of 2 element(s).
color	go_layout_smith_imaginaryaxis(), go_layout_smith_realaxis(): Sets default for all colors associated with this axis all at once: line, font, tick, and grid colors. Grid color is lightened by blending this with the plot background Individual pieces can override this. Color: a CSS color string or R color name. go_layout_smith_imaginaryaxis_tickfont(),go_layout_smith_realaxis_tickfont(): Color: a CSS color string or R color name.
gridcolor	Sets the color of the grid lines. Color: a CSS color string or R color name.
griddash	Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$".
gridwidth	Sets the width (in px) of the grid lines. Number in the interval [0, Inf].
hoverformat	Sets the hover text formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.
labelalias	Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.
layer	Sets the layer on which this axis is displayed. If "above traces", this axis is displayed above all the subplot's traces If "below traces", this axis is displayed below all the subplot's traces, but above the grid lines. Useful when used together with scatter-like traces with 'cliponaxis' set to FALSE to show markers and/or text nodes above this axis. One of "above traces", "below traces".
linecolor	Sets the axis line color. Color: a CSS color string or R color name.
linewidth	Sets the width (in px) of the axis line. Number in the interval [0, Inf].
showgrid	Determines whether or not grid lines are drawn. If TRUE, the grid lines are drawn at every tick mark. Logical.
showline	Determines whether or not a line bounding this axis is drawn. Logical.
showticklabels	Determines whether or not the tick labels are drawn. Logical.
showtickprefix	If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".
showticksuffix	Same as 'showtickprefix' but for tick suffixes. One of "all", "first", "last", "none".

tickcolor
: Sets the tick color. Color: a CSS color string or R color name.

tickfont
: go_layout_smith_imaginaryaxis(): Sets the tick font. Named list or go_layout_smith_imaginarya object.
go_layout_smith_realaxis(): Sets the tick font. Named list or go_layout_smith_realaxis_tickfo object.

tickformat
: Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

ticklen
: Sets the tick length (in px). Number in the interval [0, Inf].

tickprefix
: Sets a tick label prefix. String.

ticks
: go_layout_smith_imaginaryaxis(): Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of "outside", "inside", "".
go_layout_smith_realaxis(): Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "top" ("bottom"), this axis' are drawn above (below) the axis line. One of "top", "bottom", "".

ticksuffix
: Sets a tick label suffix. String.

tickvals
: go_layout_smith_imaginaryaxis(): Sets the values at which ticks on this axis appear. Defaults to 'realaxis.tickvals' plus the same as negatives and zero. Data array: a vector, factor, Date/POSIXct vector, matrix or list.
go_layout_smith_realaxis(): Sets the values at which ticks on this axis appear. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth
: Sets the tick width (in px). Number in the interval [0, Inf].

visible
: A single toggle to hide the axis while preserving interaction like dragging. Default is true when a cheater plot is present on the axis, otherwise false Logical.

family
: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow
: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size
: Number in the interval [1, Inf].

style
: Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".

textcase
: Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".

variant	Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
weight	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
side	Determines on which side of real axis line the tick and tick labels appear. One of "top", "bottom".
tickangle	Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

Details

go_layout_smith() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Nested objects documented here:

go_layout_smith_domain() layout.smith.domain

go_layout_smith_imaginaryaxis() layout.smith.imaginaryaxis

go_layout_smith_imaginaryaxis_tickfont() layout.smith.imaginaryaxis.tickfont: Sets the tick font.

go_layout_smith_realaxis() layout.smith.realaxis

go_layout_smith_realaxis_tickfont() layout.smith.realaxis.tickfont: Sets the tick font.

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

---

go_layout_ternary *Layout ternary objects*

Description

Properties of layout.ternary.

Usage

```
go_layout_ternary(
  arg = NULL,
  ...,
  aaxis = NULL,
  baxis = NULL,
  bgcolor = NULL,
  caxis = NULL,
  domain = NULL,
  sum = NULL,
  uirevision = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_aaxis(
  arg = NULL,
  ...,
  color = NULL,
  dtick = NULL,
  exponentformat = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
  hoverformat = NULL,
  labelalias = NULL,
  layer = NULL,
  linecolor = NULL,
  linewidth = NULL,
  min = NULL,
  minexponent = NULL,
  nticks = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showgrid = NULL,
  showline = NULL,
  showspikes = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  spikecolor = NULL,
  spikedash = NULL,
  spikemode = NULL,
  spikesnap = NULL,
  spikethickness = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
```

```
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  uirevision = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_aaxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_aaxis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_aaxis_title(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
```

```
)

go_layout_ternary_aaxis_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_baxis(
  arg = NULL,
  ...,
  color = NULL,
  dtick = NULL,
  exponentformat = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
  hoverformat = NULL,
  labelalias = NULL,
  layer = NULL,
  linecolor = NULL,
  linewidth = NULL,
  min = NULL,
  minexponent = NULL,
  nticks = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showgrid = NULL,
  showline = NULL,
  showspikes = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  spikecolor = NULL,
  spikedash = NULL,
  spikemode = NULL,
  spikesnap = NULL,
  spikethickness = NULL,
  tick0 = NULL,
```

```
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  uirevision = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_baxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_baxis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_baxis_title(
  arg = NULL,
  ...,
```

```
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_baxis_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_caxis(
  arg = NULL,
  ...,
  color = NULL,
  dtick = NULL,
  exponentformat = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
  hoverformat = NULL,
  labelalias = NULL,
  layer = NULL,
  linecolor = NULL,
  linewidth = NULL,
  min = NULL,
  minexponent = NULL,
  nticks = NULL,
  separatethousands = NULL,
  showexponent = NULL,
  showgrid = NULL,
  showline = NULL,
  showspikes = NULL,
  showticklabels = NULL,
  showtickprefix = NULL,
  showticksuffix = NULL,
  spikecolor = NULL,
  spikedash = NULL,
  spikemode = NULL,
```

```
  spikesnap = NULL,
  spikethickness = NULL,
  tick0 = NULL,
  tickangle = NULL,
  tickcolor = NULL,
  tickfont = NULL,
  tickformat = NULL,
  tickformatstops = NULL,
  ticklabelstep = NULL,
  ticklen = NULL,
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  uirevision = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_caxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_caxis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)
```

```
go_layout_ternary_caxis_title(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_caxis_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_ternary_domain(
  arg = NULL,
  ...,
  column = NULL,
  row = NULL,
  x = NULL,
  y = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
aaxis	Named list or `go_layout_ternary_aaxis()` object.
baxis	Named list or `go_layout_ternary_baxis()` object.
bgcolor	Set the background color of the subplot Color: a CSS color string or R color name.
caxis	Named list or `go_layout_ternary_caxis()` object.
domain	Named list or `go_layout_ternary_domain()` object.

sum — The number each triplet should sum to, and the maximum range of each axis Number in the interval [0, Inf].

uirevision — go_layout_ternary(): Controls persistence of user-driven changes in axis ‘min‘ and ‘title‘, if not overridden in the individual axes. Defaults to ‘layout.uirevision‘. Any value.

go_layout_ternary_aaxis(),go_layout_ternary_baxis(),go_layout_ternary_caxis(): Controls persistence of user-driven changes in axis ‘min‘, and ‘title‘ if in ‘editable: true‘ configuration. Defaults to ‘ternary<N>.uirevision‘. Any value.

skip_invalid — If TRUE, silently drop invalid properties instead of raising an error.

color — go_layout_ternary_aaxis(),go_layout_ternary_baxis(),go_layout_ternary_caxis(): Sets default for all colors associated with this axis all at once: line, font, tick, and grid colors. Grid color is lightened by blending this with the plot background Individual pieces can override this. Color: a CSS color string or R color name.

go_layout_ternary_aaxis_tickfont(),go_layout_ternary_aaxis_title_font(), go_layout_ternary_baxis_tickfont(),go_layout_ternary_baxis_title_font(), go_layout_ternary_caxis_tickfont(),go_layout_ternary_caxis_title_font(): Color: a CSS color string or R color name.

dtick — Sets the step in-between ticks on this axis. Use with ‘tick0‘. Must be a positive number, or special strings available to "log" and "date" axes. If the axis ‘type‘ is "log", then ticks are set every 10^(n*dtick) where n is the tick number. For example, to set a tick mark at 1, 10, 100, 1000, ... set dtick to 1. To set tick marks at 1, 100, 10000, ... set dtick to 2. To set tick marks at 1, 5, 25, 125, 625, 3125, ... set dtick to log_10(5), or 0.69897000433. "log" has several special values; "L<f>", where ‘f‘ is a positive number, gives ticks linearly spaced in value (but not position). For example ‘tick0‘ = 0.1, ‘dtick‘ = "L0.5" will put ticks at 0.1, 0.6, 1.1, 1.6 etc. To show powers of 10 plus small digits between, use "D1" (all digits) or "D2" (only 2 and 5). ‘tick0‘ is ignored for "D1" and "D2". If the axis ‘type‘ is "date", then you must convert the time to milliseconds. For example, to set the interval between ticks to one day, set ‘dtick‘ to 86400000.0. "date" also has special values "M<n>" gives ticks spaced by a number of months. ‘n‘ must be a positive integer. To set ticks on the 15th of every third month, set ‘tick0‘ to "2000-01-15" and ‘dtick‘ to "M3". To set ticks every 4 years, set ‘dtick‘ to "M48" Any value.

exponentformat — Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of "none", "e", "E", "power", "SI", "B", "SI extended".

gridcolor — Sets the color of the grid lines. Color: a CSS color string or R color name.

griddash — Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$".

gridwidth Sets the width (in px) of the grid lines. Number in the interval [0, Inf].

hoverformat Sets the hover text formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

labelalias Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

layer Sets the layer on which this axis is displayed. If "above traces", this axis is displayed above all the subplot's traces If "below traces", this axis is displayed below all the subplot's traces, but above the grid lines. Useful when used together with scatter-like traces with 'cliponaxis' set to FALSE to show markers and/or text nodes above this axis. One of "above traces", "below traces".

linecolor Sets the axis line color. Color: a CSS color string or R color name.

linewidth Sets the width (in px) of the axis line. Number in the interval [0, Inf].

min The minimum value visible on this axis. The maximum is determined by the sum minus the minimum values of the other two axes. The full view corresponds to all the minima set to zero. Number in the interval [0, Inf].

minexponent Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

nticks Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [1, Inf].

separatethousands If "true", even 4-digit integers are separated Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showgrid Determines whether or not grid lines are drawn. If TRUE, the grid lines are drawn at every tick mark. Logical.

showline Determines whether or not a line bounding this axis is drawn. Logical.

showspikes Determines whether or not spikes (aka droplines) are drawn for this axis. Ternary spikes are accompanied by axis value labels formatted according to 'hoverformat'. Logical.

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

showticksuffix Same as 'showtickprefix' but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

spikecolor Sets the spike color. If undefined, will use the series color Color: a CSS color string or R color name.

spikedash Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

spikemode Determines the drawing mode for the spike line If "toaxis", the line is drawn from the data point to the axis the series is plotted on. If "across", the line is drawn across the entire plot area, and supercedes "toaxis". If "marker", then a marker dot is drawn on the axis the series is plotted on Any combination of `"toaxis"`, `"across"`, `"marker"` joined with `"+"`.

spikesnap Determines whether spikelines are stuck to the cursor or to the closest datapoints. One of `"data"`, `"cursor"`, `"hovered data"`.

spikethickness Sets the spike line width in pixels. Number.

tick0 Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont `go_layout_ternary_aaxis()`: Sets the tick font. Named list or go_layout_ternary_aaxis_tickfont object.

`go_layout_ternary_baxis()`: Sets the tick font. Named list or go_layout_ternary_baxis_tickfont object.

`go_layout_ternary_caxis()`: Sets the tick font. Named list or go_layout_ternary_caxis_tickfont object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops

`go_layout_ternary_aaxis()`: List of named lists or go_layout_ternary_aaxis_tickformatstop() objects.

go_layout_ternary_baxis(): List of named lists or go_layout_ternary_baxis_tickformatstop() objects.

go_layout_ternary_caxis(): List of named lists or go_layout_ternary_caxis_tickformatstop() objects.

ticklabelstep — Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

ticklen — Sets the tick length (in px). Number in the interval [0, Inf].

tickmode — Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of "auto", "linear", "array".

tickprefix — Sets a tick label prefix. String.

ticks — Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of "outside", "inside", "".

ticksuffix — Sets a tick label suffix. String.

ticktext — Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals — Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth — Sets the tick width (in px). Number in the interval [0, Inf].

title — go_layout_ternary_aaxis(): String (the title text), named list or go_layout_ternary_aaxis_title(object.

go_layout_ternary_baxis(): String (the title text), named list or go_layout_ternary_baxis_title(object.

go_layout_ternary_caxis(): String (the title text), named list or go_layout_ternary_caxis_title(object.

family — HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String.

lineposition — Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow — Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

`size` Number in the interval [1, Inf].

`style` Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

`textcase` Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.

`variant` Sets the variant of the font. One of `"normal"`, `"small-caps"`, `"all-small-caps"`, `"all-petite-caps"`, `"petite-caps"`, `"unicase"`.

`weight` Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].

`dtickrange` range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).

`enabled` Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

`name` When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`templateitemname` Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`value` string - dtickformat for described zoom level, the same as "tickformat" String.

`font` `go_layout_ternary_aaxis_title()`: Sets this axis’ title font. Named list or `go_layout_ternary_aaxis_title_font()` object.

`go_layout_ternary_baxis_title()`: Sets this axis’ title font. Named list or `go_layout_ternary_baxis_title_font()` object.

`go_layout_ternary_caxis_title()`: Sets this axis’ title font. Named list or `go_layout_ternary_caxis_title_font()` object.

`text` Sets the title of this axis. String.

`column` If there is a layout grid, use the domain for this column in the grid for this ternary subplot. Integer in the interval [0, Inf].

`row` If there is a layout grid, use the domain for this row in the grid for this ternary subplot. Integer in the interval [0, Inf].

`x` Sets the horizontal domain of this ternary subplot (in plot fraction). Vector or list of 2 element(s).

`y` Sets the vertical domain of this ternary subplot (in plot fraction). Vector or list of 2 element(s).

Details

`go_layout_ternary()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`).

Nested objects documented here:

`go_layout_ternary_aaxis()` `layout.ternary.aaxis`

`go_layout_ternary_aaxis_tickfont()` `layout.ternary.aaxis.tickfont`: Sets the tick font.

`go_layout_ternary_aaxis_tickformatstop()` `layout.ternary.aaxis.tickformatstop`

`go_layout_ternary_aaxis_title()` `layout.ternary.aaxis.title`

`go_layout_ternary_aaxis_title_font()` `layout.ternary.aaxis.title.font`: Sets this axis' title font.

`go_layout_ternary_baxis()` `layout.ternary.baxis`

`go_layout_ternary_baxis_tickfont()` `layout.ternary.baxis.tickfont`: Sets the tick font.

`go_layout_ternary_baxis_tickformatstop()` `layout.ternary.baxis.tickformatstop`

`go_layout_ternary_baxis_title()` `layout.ternary.baxis.title`

`go_layout_ternary_baxis_title_font()` `layout.ternary.baxis.title.font`: Sets this axis' title font.

`go_layout_ternary_caxis()` `layout.ternary.caxis`

`go_layout_ternary_caxis_tickfont()` `layout.ternary.caxis.tickfont`: Sets the tick font.

`go_layout_ternary_caxis_tickformatstop()` `layout.ternary.caxis.tickformatstop`

`go_layout_ternary_caxis_title()` `layout.ternary.caxis.title`

`go_layout_ternary_caxis_title_font()` `layout.ternary.caxis.title.font`: Sets this axis' title font.

`go_layout_ternary_domain()` `layout.ternary.domain`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

go_layout_title *Layout title objects*

Description

Properties of `layout.title`.

Usage

```
go_layout_title(
  arg = NULL,
  ...,
  automargin = NULL,
  font = NULL,
  pad = NULL,
  subtitle = NULL,
  text = NULL,
  x = NULL,
  xanchor = NULL,
  xref = NULL,
  y = NULL,
  yanchor = NULL,
  yref = NULL,
  skip_invalid = FALSE
)

go_layout_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_title_pad(
  arg = NULL,
  ...,
  b = NULL,
  l = NULL,
  r = NULL,
```

```
  t = NULL,
  skip_invalid = FALSE
)

go_layout_title_subtitle(
  arg = NULL,
  ...,
  font = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_layout_title_subtitle_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
automargin	Determines whether the title can automatically push the figure margins. If 'yref='paper'' then the margin will expand to ensure that the title doesn't overlap with the edges of the container. If 'yref='container'' then the margins will ensure that the title doesn't overlap with the plot area, tick labels, and axis titles. If 'automargin=true' and the margins need to be expanded, then y will be set to a default 1 and yanchor will be set to an appropriate default to ensure that minimal margin space is needed. Note that when 'yref='paper'', only 1 or 0 are allowed y values. Invalid values will be reset to the default 1. Logical.
font	`go_layout_title()`: Sets the title font. Named list or `go_layout_title_font()` object. `go_layout_title_subtitle()`: Sets the subtitle font. Named list or `go_layout_title_subtitle_font` object.
pad	Sets the padding of the title. Each padding value only applies when the corresponding 'xanchor'/'yanchor' value is set accordingly. E.g. for left padding

	to take effect, ‘xanchor‘ must be set to "left". The same rule applies if ‘xanchor‘/‘yanchor‘ is determined automatically. Padding is muted if the respective anchor value is "middle"/"center". Named list or `go_layout_title_pad()` object.
`subtitle`	Named list or `go_layout_title_subtitle()` object.
`text`	`go_layout_title()`: Sets the plot's title. String. `go_layout_title_subtitle()`: Sets the plot's subtitle. String.
`x`	Sets the x position with respect to ‘xref‘ in normalized coordinates from 0 (left) to 1 (right). Number in the interval [0, 1].
`xanchor`	Sets the title's horizontal alignment with respect to its x position. "left" means that the title starts at x, "right" means that the title ends at x and "center" means that the title's center is at x. "auto" divides ‘xref‘ by three and calculates the ‘xanchor‘ value automatically based on the value of ‘x‘. One of `"auto"`, `"left"`, `"center"`, `"right"`.
`xref`	Sets the container ‘x‘ refers to. "container" spans the entire ‘width‘ of the plot. "paper" refers to the width of the plotting area only. One of `"container"`, `"paper"`.
`y`	Sets the y position with respect to ‘yref‘ in normalized coordinates from 0 (bottom) to 1 (top). "auto" places the baseline of the title onto the vertical center of the top margin. Number in the interval [0, 1].
`yanchor`	Sets the title's vertical alignment with respect to its y position. "top" means that the title's cap line is at y, "bottom" means that the title's baseline is at y and "middle" means that the title's midline is at y. "auto" divides ‘yref‘ by three and calculates the ‘yanchor‘ value automatically based on the value of ‘y‘. One of `"auto"`, `"top"`, `"middle"`, `"bottom"`.
`yref`	Sets the container ‘y‘ refers to. "container" spans the entire ‘height‘ of the plot. "paper" refers to the height of the plotting area only. One of `"container"`, `"paper"`.
`skip_invalid`	If `TRUE`, silently drop invalid properties instead of raising an error.
`color`	Color: a CSS color string or R color name.
`family`	HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.
`lineposition`	Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.
`shadow`	Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.
`size`	Number in the interval [1, Inf].
`style`	Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.

textcase	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".
variant	Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
weight	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
b	The amount of padding (in px) along the bottom of the component. Number.
l	The amount of padding (in px) on the left side of the component. Number.
r	The amount of padding (in px) on the right side of the component. Number.
t	The amount of padding (in px) along the top of the component. Number.

Details

go_layout_title() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Nested objects documented here:

go_layout_title_font() layout.title.font: Sets the title font.

go_layout_title_pad() layout.title.pad: Sets the padding of the title. Each padding value only applies when the corresponding ‘xanchor‘/‘yanchor‘ value is set accordingly. E.g. for left padding to take effect, ‘xanchor‘ must be set to "left". The same rule applies if ‘xanchor‘/‘yanchor‘ is determined automatically. Padding is muted if the respective anchor value is "middle"/"center".

go_layout_title_subtitle() layout.title.subtitle

go_layout_title_subtitle_font() layout.title.subtitle.font: Sets the subtitle font.

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

---

go_layout_transition *Layout transition objects*

Description

Sets transition options used during Plotly.react updates.

Usage

```
go_layout_transition(
  arg = NULL,
  ...,
  duration = NULL,
  easing = NULL,
  ordering = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. marker_size = 10 for marker = list(size = 10). Property values may be one-sided formulas, evaluated in data for add_*() functions.
duration	The duration of the transition, in milliseconds. If equal to zero, updates are synchronous. Number in the interval [0, Inf].
easing	The easing function used for the transition One of "linear", "quad", "cubic", "sin", "exp", "circle", "elastic", "back", "bounce", "linear-in", "quad-in", "cubic-in", "sin-in", "exp-in", "circle-in", "elastic-in", "back-in", "bounce-in", "linear-out", "quad-out", "cubic-out", "sin-out", "exp-out", "circle-out", "elastic-out", "back-out", "bounce-out", "linear-in-out", "quad-in-out", "cubic-in-out", "sin-in-out", "exp-in-out", "circle-in-out", "elastic-in-out", "back-in-out", "bounce-in-out".
ordering	Determines whether the figure's layout or traces smoothly transitions during updates that make both traces and layout change. One of "layout first", "traces first".
skip_invalid	If TRUE, silently drop invalid properties instead of raising an error.

Details

go_layout_transition() create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Value

Each go_*() function returns a graph object of the same class as its name; add_*() functions return the modified figure.

See Also

go_layout()

`go_layout_uniformtext` *Layout uniformtext objects*

Description

Properties of `layout.uniformtext`.

Usage

```
go_layout_uniformtext(
  arg = NULL,
  ...,
  minsize = NULL,
  mode = NULL,
  skip_invalid = FALSE
)
```

Arguments

`arg`	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
`...`	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
`minsize`	Sets the minimum text size between traces of the same type. Number in the interval [0, Inf].
`mode`	Determines how the font size for various text elements are uniformed between each trace type. If the computed text sizes were smaller than the minimum size defined by ‘uniformtext.minsize‘ using "hide" option hides the text; and using "show" option shows the text without further downscaling. Please note that if the size defined by ‘minsize‘ is greater than the font size defined by trace, then the ‘minsize‘ is used. One of `"hide"`, `"show"`, `FALSE`.
`skip_invalid`	If `TRUE`, silently drop invalid properties instead of raising an error.

Details

`go_layout_uniformtext()` create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`).

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

go_layout_updatemenu *Layout updatemenu objects*

Description

Properties of `layout.updatemenu`.

Usage

```
go_layout_updatemenu(
  arg = NULL,
  ...,
  active = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  buttons = NULL,
  direction = NULL,
  font = NULL,
  name = NULL,
  pad = NULL,
  showactive = NULL,
  templateitemname = NULL,
  type = NULL,
  visible = NULL,
  x = NULL,
  xanchor = NULL,
  y = NULL,
  yanchor = NULL,
  skip_invalid = FALSE
)

go_layout_updatemenu_button(
  arg = NULL,
  ...,
  args = NULL,
  args2 = NULL,
  execute = NULL,
  label = NULL,
  method = NULL,
  name = NULL,
  templateitemname = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_layout_updatemenu_font(
```

```
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_updatemenu_pad(
  arg = NULL,
  ...,
  b = NULL,
  l = NULL,
  r = NULL,
  t = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
active	Determines which button (by index starting from 0) is considered active. Integer in the interval [-1, Inf].
bgcolor	Sets the background color of the update menu buttons. Color: a CSS color string or R color name.
bordercolor	Sets the color of the border enclosing the update menu. Color: a CSS color string or R color name.
borderwidth	Sets the width (in px) of the border enclosing the update menu. Number in the interval [0, Inf].
buttons	List of named lists or `go_layout_updatemenu_button()` objects.
direction	Determines the direction in which the buttons are laid out, whether in a dropdown menu or a row/column of buttons. For 'left' and 'up', the buttons will still appear in left-to-right or top-to-bottom order respectively. One of `"left"`, `"right"`, `"up"`, `"down"`.
font	Sets the font of the update menu button text. Named list or `go_layout_updatemenu_font()` object.

`name` When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`pad` Sets the padding around the buttons or dropdown menu. Named list or `go_layout_updatemenu_pad()` object.

`showactive` Highlights active dropdown item or active button if true. Logical.

`templateitemname` Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`type` Determines whether the buttons are accessible via a dropdown menu or whether the buttons are stacked horizontally or vertically One of `"dropdown"`, `"buttons"`.

`visible` `go_layout_updatemenu()`: Determines whether or not the update menu is visible. Logical.

`go_layout_updatemenu_button()`: Determines whether or not this button is visible. Logical.

`x` Sets the x position (in normalized coordinates) of the update menu. Number in the interval [-2, 3].

`xanchor` Sets the update menu’s horizontal position anchor. This anchor binds the ‘x‘ position to the "left", "center" or "right" of the range selector. One of `"auto"`, `"left"`, `"center"`, `"right"`.

`y` Sets the y position (in normalized coordinates) of the update menu. Number in the interval [-2, 3].

`yanchor` Sets the update menu’s vertical position anchor This anchor binds the ‘y‘ position to the "top", "middle" or "bottom" of the range selector. One of `"auto"`, `"top"`, `"middle"`, `"bottom"`.

`skip_invalid` If `TRUE`, silently drop invalid properties instead of raising an error.

`args` Sets the arguments values to be passed to the Plotly method set in ‘method‘ on click. Vector or list of up to 3 element(s).

`args2` Sets a 2nd set of ‘args‘, these arguments values are passed to the Plotly method set in ‘method‘ when clicking this button while in the active state. Use this to create toggle buttons. Vector or list of up to 3 element(s).

`execute` When true, the API method is executed. When false, all other behaviors are the same and command execution is skipped. This may be useful when hooking into, for example, the ‘plotly_buttonclicked‘ method and executing the API command manually without losing the benefit of the updatemenu automatically binding to the state of the plot through the specification of ‘method‘ and ‘args‘. Logical.

`label` Sets the text label to appear on the button. String.

method	Sets the Plotly method to be called on click. If the 'skip' method is used, the API updatemenu will function as normal but will perform no API calls and will not bind automatically to state updates. This may be used to create a component interface and attach to updatemenu events manually via JavaScript. One of "restyle", "relayout", "animate", "update", "skip".
color	Color: a CSS color string or R color name.
family	HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.
lineposition	Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".
shadow	Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.
size	Number in the interval [1, Inf].
style	Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".
textcase	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".
variant	Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
weight	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
b	The amount of padding (in px) along the bottom of the component. Number.
l	The amount of padding (in px) on the left side of the component. Number.
r	The amount of padding (in px) on the right side of the component. Number.
t	The amount of padding (in px) along the top of the component. Number.

Details

go_layout_updatemenu() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Nested objects documented here:

go_layout_updatemenu_button() layout.updatemenu.button

go_layout_updatemenu_font() layout.updatemenu.font: Sets the font of the update menu button text.

go_layout_updatemenu_pad() layout.updatemenu.pad: Sets the padding around the buttons or dropdown menu.

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

`go_layout_xaxis` *Layout xaxis objects*

Description

Properties of `layout.xaxis`.

Usage

```
go_layout_xaxis(
  arg = NULL,
  ...,
  anchor = NULL,
  automargin = NULL,
  autorange = NULL,
  autorangeoptions = NULL,
  autotickangles = NULL,
  autotypenumbers = NULL,
  calendar = NULL,
  categoryarray = NULL,
  categoryorder = NULL,
  color = NULL,
  constrain = NULL,
  constraintoward = NULL,
  dividercolor = NULL,
  dividerwidth = NULL,
  domain = NULL,
  domainpad = NULL,
  dtick = NULL,
  exponentformat = NULL,
  fixedrange = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
  hoverformat = NULL,
  insiderange = NULL,
  labelalias = NULL,
  layer = NULL,
  linecolor = NULL,
```

```
    linewidth = NULL,
    matches = NULL,
    maxallowed = NULL,
    minallowed = NULL,
    minexponent = NULL,
    minor = NULL,
    minorloglabels = NULL,
    mirror = NULL,
    modebardisable = NULL,
    nticks = NULL,
    overlaying = NULL,
    position = NULL,
    range = NULL,
    rangebreaks = NULL,
    rangemode = NULL,
    rangeselector = NULL,
    rangeslider = NULL,
    scaleanchor = NULL,
    scaleratio = NULL,
    separatethousands = NULL,
    showdividers = NULL,
    showexponent = NULL,
    showgrid = NULL,
    showline = NULL,
    showspikes = NULL,
    showticklabels = NULL,
    showtickprefix = NULL,
    showticksuffix = NULL,
    side = NULL,
    spikecolor = NULL,
    spikedash = NULL,
    spikemode = NULL,
    spikesnap = NULL,
    spikethickness = NULL,
    tick0 = NULL,
    tickangle = NULL,
    tickcolor = NULL,
    tickfont = NULL,
    tickformat = NULL,
    tickformatstops = NULL,
    ticklabelindex = NULL,
    ticklabelmode = NULL,
    ticklabeloverflow = NULL,
    ticklabelposition = NULL,
    ticklabelshift = NULL,
    ticklabelstandoff = NULL,
    ticklabelstep = NULL,
    ticklen = NULL,
```

```
  tickmode = NULL,
  tickprefix = NULL,
  ticks = NULL,
  tickson = NULL,
  ticksuffix = NULL,
  ticktext = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  title = NULL,
  type = NULL,
  uirevision = NULL,
  unifiedhovertitle = NULL,
  visible = NULL,
  zeroline = NULL,
  zerolinecolor = NULL,
  zerolinelayer = NULL,
  zerolinewidth = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_autorangeoptions(
  arg = NULL,
  ...,
  clipmax = NULL,
  clipmin = NULL,
  include = NULL,
  maxallowed = NULL,
  minallowed = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_domainpad(
  arg = NULL,
  ...,
  bottom = NULL,
  left = NULL,
  right = NULL,
  top = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_minor(
  arg = NULL,
  ...,
  dtick = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
```

```
  nticks = NULL,
  showgrid = NULL,
  tick0 = NULL,
  tickcolor = NULL,
  ticklen = NULL,
  tickmode = NULL,
  ticks = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_rangebreak(
  arg = NULL,
  ...,
  bounds = NULL,
  dvalue = NULL,
  enabled = NULL,
  name = NULL,
  pattern = NULL,
  templateitemname = NULL,
  values = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_rangeselector(
  arg = NULL,
  ...,
  activecolor = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  buttons = NULL,
  font = NULL,
  visible = NULL,
  x = NULL,
  xanchor = NULL,
  y = NULL,
  yanchor = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_rangeselector_button(
  arg = NULL,
  ...,
  count = NULL,
  label = NULL,
  name = NULL,
```

```
  step = NULL,
  stepmode = NULL,
  templateitemname = NULL,
  visible = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_rangeselector_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_rangeslider(
  arg = NULL,
  ...,
  autorange = NULL,
  bgcolor = NULL,
  bordercolor = NULL,
  borderwidth = NULL,
  range = NULL,
  thickness = NULL,
  visible = NULL,
  yaxis = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_rangeslider_yaxis(
  arg = NULL,
  ...,
  range = NULL,
  rangemode = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
```

```
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_title(
  arg = NULL,
  ...,
  font = NULL,
  standoff = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_xaxis_unifiedhovertitle(
  arg = NULL,
```

```
  ...,
  text = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
anchor	If set to an opposite-letter axis id (e.g. 'x2', 'y'), this axis is bound to the corresponding opposite-letter axis. If set to "free", this axis' position is determined by 'position'. One of `"free"`, or a string matching `"^x([2-9]|[1-9][0-9]+)?( domain)?$"` or `"^y([2-9]\|[1-9][0-9]+)?( domain)?$"`.
automargin	Determines whether long tick labels automatically grow the figure margins. Any combination of `"height"`, `"width"`, `"left"`, `"right"`, `"top"`, `"bottom"` joined with `"+"`, or one of `TRUE`, `FALSE`.
autorange	`go_layout_xaxis()`: Determines whether or not the range of this axis is computed in relation to the input data. See 'rangemode' for more info. If 'range' is provided and it has a value for both the lower and upper bound, 'autorange' is set to FALSE. Using "min" applies autorange only to set the minimum. Using "max" applies autorange only to set the maximum. Using "min reversed" applies autorange only to set the minimum on a reversed axis. Using "max reversed" applies autorange only to set the maximum on a reversed axis. Using "reversed" applies autorange on both ends and reverses the axis direction. One of `"reversed"`, `"min reversed"`, `"max reversed"`, `"min"`, `"max"`, `TRUE`, `FALSE`. `go_layout_xaxis_rangeslider()`: Determines whether or not the range slider range is computed in relation to the input data. If 'range' is provided, then 'autorange' is set to FALSE. Logical.
autorangeoptions	Named list or `go_layout_xaxis_autorangeoptions()` object.
autotickangles	When 'tickangle' is set to "auto", it will be set to the first angle in this array that is large enough to prevent label overlap. Vector or list of up to 1 element(s).
autotypenumbers	Using "strict" a numeric string in trace data is not converted to a number. Using "convert types" a numeric string in trace data may be treated as a number during automatic axis 'type' detection. Defaults to layout.autotypenumbers. One of `"convert types"`, `"strict"`.
calendar	Sets the calendar system to use for 'range' and 'tick0' if this is a date axis. This does not set the calendar for interpreting data on this axis, that's specified in the trace or via the global 'layout.calendar' One of `"chinese"`, `"coptic"`, `"discworld"`, `"ethiopian"`, `"gregorian"`, `"hebrew"`, `"islamic"`, `"jalali"`, `"julian"`, `"mayan"`, `"nanakshahi"`, `"nepali"`, `"persian"`, `"taiwan"`, `"thai"`, `"ummalqura"`.

`categoryarray` Sets the order in which categories on this axis appear. Only has an effect if ‘categoryorder‘ is set to "array". Used with ‘categoryorder‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`categoryorder` Specifies the ordering logic for the case of categorical variables. By default, plotly uses "trace", which specifies the order that is present in the data supplied. Set ‘categoryorder‘ to "category ascending" or "category descending" if order should be determined by the alphanumerical order of the category names. Set ‘categoryorder‘ to "array" to derive the ordering from the attribute ‘categoryarray‘. If a category is not found in the ‘categoryarray‘ array, the sorting behavior for that attribute will be identical to the "trace" mode. The unspecified categories will follow the categories in ‘categoryarray‘. Set ‘categoryorder‘ to "total ascending" or "total descending" if order should be determined by the numerical order of the values. Similarly, the order can be determined by the min, max, sum, mean, geometric mean or median of all the values. One of `"trace"`, `"category ascending"`, `"category descending"`, `"array"`, `"total ascending"`, `"total descending"`, `"min ascending"`, `"min descending"`, `"max ascending"`, `"max descending"`, `"sum ascending"`, `"sum descending"`, `"mean ascending"`, `"mean descending"`, `"geometric mean ascending"`, `"geometric mean descending"`, `"median ascending"`, `"median descending"`.

`color` `go_layout_xaxis()`: Sets default for all colors associated with this axis all at once: line, font, tick, and grid colors. Grid color is lightened by blending this with the plot background Individual pieces can override this. Color: a CSS color string or R color name.

`go_layout_xaxis_rangeselector_font()`, `go_layout_xaxis_tickfont()`, `go_layout_xaxis_title_font()`: Color: a CSS color string or R color name.

`constrain` If this axis needs to be compressed (either due to its own ‘scaleanchor‘ and ‘scaleratio‘ or those of the other axis), determines how that happens: by increasing the "range", or by decreasing the "domain". Default is "domain" for axes containing image traces, "range" otherwise. One of `"range"`, `"domain"`.

`constraintoward` If this axis needs to be compressed (either due to its own ‘scaleanchor‘ and ‘scaleratio‘ or those of the other axis), determines which direction we push the originally specified plot area. Options are "left", "center" (default), and "right" for x axes, and "top", "middle" (default), and "bottom" for y axes. One of `"left"`, `"center"`, `"right"`, `"top"`, `"middle"`, `"bottom"`.

`dividercolor` Sets the color of the dividers Only has an effect on "multicategory" axes. Color: a CSS color string or R color name.

`dividerwidth` Sets the width (in px) of the dividers Only has an effect on "multicategory" axes. Number.

`domain` Sets the domain of this axis (in plot fraction). Vector or list of 2 element(s).

`domainpad` Sets the space in pixels to reserve inside the edges of ‘domain‘. Because ‘domain‘ is a fraction of the plot area, the gap between subplots changes with the figure size. ‘domainpad‘ keeps the same size in pixels at all figure sizes, so it can hold content with a fixed size, such as a subplot title. x axes use ‘left‘ and ‘right‘. y axes use ‘top‘ and ‘bottom‘. If the padding needs more room than the domain has, the two pads shrink by the same factor, and the plot area keeps a length of

at least 2 pixels. An axis that sets ‘overlaying‘ ignores its own ‘domainpad‘ and uses the ‘domainpad‘ of the axis it overlays, so that both axes share the same plot area. Named list or `go_layout_xaxis_domainpad()` object.

`dtick` Sets the step in-between ticks on this axis. Use with ‘tick0‘. Must be a positive number, or special strings available to "log" and "date" axes. If the axis ‘type‘ is "log", then ticks are set every 10^(n*dtick) where n is the tick number. For example, to set a tick mark at 1, 10, 100, 1000, ... set dtick to 1. To set tick marks at 1, 100, 10000, ... set dtick to 2. To set tick marks at 1, 5, 25, 125, 625, 3125, ... set dtick to log_10(5), or 0.69897000433. "log" has several special values; "L<f>", where ‘f‘ is a positive number, gives ticks linearly spaced in value (but not position). For example ‘tick0‘ = 0.1, ‘dtick‘ = "L0.5" will put ticks at 0.1, 0.6, 1.1, 1.6 etc. To show powers of 10 plus small digits between, use "D1" (all digits) or "D2" (only 2 and 5). ‘tick0‘ is ignored for "D1" and "D2". If the axis ‘type‘ is "date", then you must convert the time to milliseconds. For example, to set the interval between ticks to one day, set ‘dtick‘ to 86400000.0. "date" also has special values "M<n>" gives ticks spaced by a number of months. ‘n‘ must be a positive integer. To set ticks on the 15th of every third month, set ‘tick0‘ to "2000-01-15" and ‘dtick‘ to "M3". To set ticks every 4 years, set ‘dtick‘ to "M48" Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`fixedrange` Determines whether or not this axis is zoom-able. If true, then zoom is disabled. Logical.

`gridcolor` Sets the color of the grid lines. Color: a CSS color string or R color name.

`griddash` Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

`gridwidth` Sets the width (in px) of the grid lines. Number in the interval [0, Inf].

`hoverformat` Sets the hover text formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`insiderange` Could be used to set the desired inside range of this axis (excluding the labels) when ‘ticklabelposition‘ of the anchored axis has "inside". Not implemented for axes with ‘type‘ "log". This would be ignored when ‘range‘ is provided. Vector or list of 2 element(s).

labelalias — Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

layer — Sets the layer on which this axis is displayed. If "above traces", this axis is displayed above all the subplot's traces If "below traces", this axis is displayed below all the subplot's traces, but above the grid lines. Useful when used together with scatter-like traces with 'cliponaxis' set to FALSE to show markers and/or text nodes above this axis. One of `"above traces"`, `"below traces"`.

linecolor — Sets the axis line color. Color: a CSS color string or R color name.

linewidth — Sets the width (in px) of the axis line. Number in the interval [0, Inf].

matches — If set to another axis id (e.g. 'x2', 'y'), the range of this axis will match the range of the corresponding axis in data-coordinates space. Moreover, matching axes share auto-range values, category lists and histogram auto-bins. Note that setting axes simultaneously in both a 'scaleanchor' and a 'matches' constraint is currently forbidden. Moreover, note that matching axes must have the same 'type'. One of , or a string matching `"^x([2-9]|[1-9][0-9]+)?( domain)?$"` or `"^y([2-9]|[1-9][0-9]+)?( domain)?$"`.

maxallowed — `go_layout_xaxis()`: Determines the maximum range of this axis. Any value.
`go_layout_xaxis_autorangeoptions()`: Use this value exactly as autorange maximum. Any value.

minallowed — `go_layout_xaxis()`: Determines the minimum range of this axis. Any value.
`go_layout_xaxis_autorangeoptions()`: Use this value exactly as autorange minimum. Any value.

minexponent — Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

minor — Named list or `go_layout_xaxis_minor()` object.

minorloglabels — Determines how minor log labels are displayed. If "small digits", small digits i.e. 2 or 5 are displayed. If "complete", complete digits are displayed. If "none", no labels are displayed. One of `"small digits"`, `"complete"`, `"none"`.

mirror — Determines if the axis lines or/and ticks are mirrored to the opposite side of the plotting area. If TRUE, the axis lines are mirrored. If "ticks", the axis lines and ticks are mirrored. If FALSE, mirroring is disable. If "all", axis lines are mirrored on all shared-axes subplots. If "allticks", axis lines and ticks are mirrored on all shared-axes subplots. One of `"ticks"`, `"all"`, `"allticks"`, `TRUE`, `FALSE`.

modebardisable — Disables certain modebar buttons for this axis. "autoscale" disables the autoscale buttons, "zoominout" disables the zoom-in and zoom-out buttons. Any combination of `"autoscale"`, `"zoominout"` joined with `"+"`, or one of `"none"`.

nticks — Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to 'nticks'. Has an effect only if 'tickmode' is set to "auto". Integer in the interval [0, Inf].

overlaying — If set a same-letter axis id, this axis is overlaid on top of the corresponding same-letter axis, with traces and axes visible for both axes. If FALSE, this axis does not overlay any same-letter axes. In this case, for axes with overlapping domains only the highest-numbered axis will be visible. One of `"free"`, or a string matching `"^x([2-9]|[1-9][0-9]+)?( domain)?$"` or `"^y([2-9]|[1-9][0-9]+)?( domain)?$"`.

position — Sets the position of this axis in the plotting space (in normalized coordinates). Only has an effect if ‘anchor‘ is set to "free". Number in the interval [0, 1].

range — `go_layout_xaxis()`: Sets the range of this axis. If the axis ‘type‘ is "log", then you must take the log of your desired range (e.g. to set the range from 1 to 100, set the range from 0 to 2). If the axis ‘type‘ is "date", it should be date strings, like date data, though Date objects and unix milliseconds will be accepted and converted to strings. If the axis ‘type‘ is "category", it should be numbers, using the scale where each category is assigned a serial number from zero in the order it appears. Leaving either or both elements ‘null‘ impacts the default ‘autorange‘. Vector or list of 2 element(s).

`go_layout_xaxis_rangeslider()`: Sets the range of the range slider. If not set, defaults to the full xaxis range. If the axis ‘type‘ is "log", then you must take the log of your desired range. If the axis ‘type‘ is "date", it should be date strings, like date data, though Date objects and unix milliseconds will be accepted and converted to strings. If the axis ‘type‘ is "category", it should be numbers, using the scale where each category is assigned a serial number from zero in the order it appears. Vector or list of 2 element(s).

`go_layout_xaxis_rangeslider_yaxis()`: Sets the range of this axis for the rangeslider. Vector or list of 2 element(s).

rangebreaks — List of named lists or `go_layout_xaxis_rangebreak()` objects.

rangemode — `go_layout_xaxis()`: If "normal", the range is computed in relation to the extrema of the input data. If "tozero", the range extends to 0, regardless of the input data If "nonnegative", the range is non-negative, regardless of the input data. Applies only to linear axes. One of `"normal"`, `"tozero"`, `"nonnegative"`.

`go_layout_xaxis_rangeslider_yaxis()`: Determines whether or not the range of this axis in the rangeslider use the same value than in the main plot when zooming in/out. If "auto", the autorange will be used. If "fixed", the ‘range‘ is used. If "match", the current range of the corresponding y-axis on the main subplot is used. One of `"auto"`, `"fixed"`, `"match"`.

rangeselector — Named list or `go_layout_xaxis_rangeselector()` object.

rangeslider — Named list or `go_layout_xaxis_rangeslider()` object.

scaleanchor — If set to another axis id (e.g. ‘x2‘, ‘y‘), the range of this axis changes together with the range of the corresponding axis such that the scale of pixels per unit is in a constant ratio. Both axes are still zoomable, but when you zoom one, the other will zoom the same amount, keeping a fixed midpoint. ‘constrain‘ and ‘constraintoward‘ determine how we enforce the constraint. You can chain these, ie ‘yaxis: {scaleanchor: "x"}, xaxis2: {scaleanchor: "y"}‘ but you can only link axes of the same ‘type‘. The linked axis can have the opposite letter (to constrain the aspect ratio) or the same letter (to match scales across subplots). Loops (‘yaxis: {scaleanchor: "x"}, xaxis: {scaleanchor: "y"}‘ or longer) are redundant

and the last constraint encountered will be ignored to avoid possible inconsistent constraints via ‘scaleratio‘. Note that setting axes simultaneously in both a ‘scaleanchor‘ and a ‘matches‘ constraint is currently forbidden. Setting ‘false‘ allows to remove a default constraint (occasionally, you may need to prevent a default ‘scaleanchor‘ constraint from being applied, eg. when having an image trace ‘yaxis: {scaleanchor: "x"}‘ is set automatically in order for pixels to be rendered as squares, setting ‘yaxis: {scaleanchor: false}‘ allows to remove the constraint). One of `FALSE`, or a string matching `"^x([2-9]|[1-9][0-9]+)?( domain)?$"` or `"^y([2-9]|[1-9][0-9]+)?( domain)?$"`.

`scaleratio` If this axis is linked to another by ‘scaleanchor‘, this determines the pixel to unit scale ratio. For example, if this value is 10, then every unit on this axis spans 10 times the number of pixels as a unit on the linked axis. Use this for example to create an elevation profile where the vertical scale is exaggerated a fixed amount with respect to the horizontal. Number in the interval [0, Inf].

`separatethousands`
If "true", even 4-digit integers are separated Logical.

`showdividers` Determines whether or not a dividers are drawn between the category levels of this axis. Only has an effect on "multicategory" axes. Logical.

`showexponent` If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showgrid` Determines whether or not grid lines are drawn. If TRUE, the grid lines are drawn at every tick mark. Logical.

`showline` Determines whether or not a line bounding this axis is drawn. Logical.

`showspikes` Determines whether or not spikes (aka droplines) are drawn for this axis. Note that spikes will never be drawn when ‘hovermode‘ is FALSE. Logical.

`showticklabels` Determines whether or not the tick labels are drawn. Logical.

`showtickprefix` If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of `"all"`, `"first"`, `"last"`, `"none"`.

`showticksuffix` Same as ‘showtickprefix‘ but for tick suffixes. One of `"all"`, `"first"`, `"last"`, `"none"`.

`side` Determines whether a x (y) axis is positioned at the "bottom" ("left") or "top" ("right") of the plotting area. One of `"top"`, `"bottom"`, `"left"`, `"right"`.

`spikecolor` Sets the spike color. If undefined, will use the series color Color: a CSS color string or R color name.

`spikedash` Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

`spikemode` Determines the drawing mode for the spike line If "toaxis", the line is drawn from the data point to the axis the series is plotted on. If "across", the line is drawn across the entire plot area, and supercedes "toaxis". If "marker", then a marker dot is drawn on the axis the series is plotted on Any combination of `"toaxis"`, `"across"`, `"marker"` joined with `"+"`.

spikesnap Determines whether spikelines are stuck to the cursor or to the closest datapoints. One of "data", "cursor", "hovered data".

spikethickness Sets the spike line width in pixels. Number.

tick0 Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the tick font. Named list or go_layout_xaxis_tickfont() object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or go_layout_xaxis_tickformatstop() objects.

ticklabelindex Only for axes with 'type' "date" or "linear". Instead of drawing the major tick label, draw the label for the minor tick that is n positions away from the major tick. E.g. to always draw the label for the minor tick before each major tick, choose 'ticklabelindex' -1. This is useful for date axes with 'ticklabelmode' "period" if you want to label the period that ends with each major tick instead of the period that begins there. Integer, or a vector of these.

ticklabelmode Determines where tick labels are drawn with respect to their corresponding ticks and grid lines. Only has an effect for axes of 'type' "date" When set to "period", tick labels are drawn in the middle of the period between ticks. One of "instant", "period".

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". Otherwise on "category" and "multicategory" axes the default is "allow". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

ticklabelposition Determines where tick labels are drawn with respect to the axis. Please note that top or bottom has no effect on x axes or when 'ticklabelmode' is set to "period" or when 'tickson' is set to "boundaries". Similarly, left or right has no effect on y axes or when 'ticklabelmode' is set to "period" or when 'tickson' is set to "boundaries". Has no effect on "multicategory" axes. When used on axes linked by 'matches' or 'scaleanchor', no extra padding for inside labels would

be added by autorange, so that the scales could match. One of "outside", "inside", "outside top", "inside top", "outside left", "inside left", "outside right", "inside right", "outside bottom", "inside bottom".

ticklabelshift Shifts the tick labels by the specified number of pixels in parallel to the axis. Positive values move the labels in the positive direction of the axis. Integer.

ticklabelstandoff

Sets the standoff distance (in px) between the axis tick labels and their default position. A positive ‘ticklabelstandoff‘ moves the labels farther away from the plot area if ‘ticklabelposition‘ is "outside", and deeper into the plot area if ‘ticklabelposition‘ is "inside". A negative ‘ticklabelstandoff‘ works in the opposite direction, moving outside ticks towards the plot area and inside ticks towards the outside. If the negative value is large enough, inside ticks can even end up outside and vice versa. Integer.

ticklabelstep Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

ticklen Sets the tick length (in px). Number in the interval [0, Inf].

tickmode `go_layout_xaxis()`: Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). If "sync", the number of ticks will sync with the overlayed axis set by ‘overlaying‘ property. When no other tick info is provided, overlaying (non-categorical) axes default to "sync", while other axes default to "auto". One of "auto", "linear", "array", "sync".

`go_layout_xaxis_minor()`: Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of "auto", "linear", "array".

tickprefix Sets a tick label prefix. String.

ticks Determines whether ticks are drawn or not. If **, this axis' ticks are not drawn. If "outside" ("inside"), this axis' are drawn outside (inside) the axis lines. One of "outside", "inside", "".

tickson Determines where ticks and grid lines are drawn with respect to their corresponding tick labels. Only has an effect for axes of ‘type‘ "category" or "multicategory". When set to "boundaries", ticks and grid lines are drawn half a category to the left/bottom of labels. One of "labels", "boundaries".

ticksuffix Sets a tick label suffix. String.

ticktext Sets the text displayed at the ticks position via ‘tickvals‘. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘tickvals‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickvals — Sets the values at which ticks on this axis appear. Only has an effect if ‘tickmode‘ is set to "array". Used with ‘ticktext‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

tickwidth — Sets the tick width (in px). Number in the interval [0, Inf].

title — String (the title text), named list or go_layout_xaxis_title() object.

type — Sets the axis type. By default, plotly attempts to determined the axis type by looking into the data of the traces that referenced the axis in question. One of "-", "linear", "log", "date", "category", "multicategory".

uirevision — Controls persistence of user-driven changes in axis ‘range‘, ‘autorange‘, and ‘title‘ if in ‘editable: true‘ configuration. Defaults to ‘layout.uirevision‘. Any value.

unifiedhovertitle — Named list or go_layout_xaxis_unifiedhovertitle() object.

visible — go_layout_xaxis(): A single toggle to hide the axis while preserving interaction like dragging. Default is true when a cheater plot is present on the axis, otherwise false Logical.

go_layout_xaxis_rangeselector(): Determines whether or not this range selector is visible. Note that range selectors are only available for x axes of ‘type‘ set to or auto-typed to "date". Logical.

go_layout_xaxis_rangeselector_button(): Determines whether or not this button is visible. Logical.

go_layout_xaxis_rangeslider(): Determines whether or not the range slider will be visible. If visible, perpendicular axes will be set to ‘fixedrange‘ Logical.

zeroline — Determines whether or not a line is drawn at along the 0 value of this axis. If TRUE, the zero line is drawn on top of the grid lines. Logical.

zerolinecolor — Sets the line color of the zero line. Color: a CSS color string or R color name.

zerolinelayer — Sets the layer on which this zeroline is displayed. If "above traces", this zeroline is displayed above all the subplot's traces If "below traces", this zeroline is displayed below all the subplot's traces, but above the grid lines. Limitation: "zerolinelayer" currently has no effect if the "zorder" property is set on any trace. One of "above traces", "below traces".

zerolinewidth — Sets the width (in px) of the zero line. Number.

skip_invalid — If TRUE, silently drop invalid properties instead of raising an error.

clipmax — Clip autorange maximum if it goes beyond this value. Has no effect when ‘autorangeoptions.maxallowed‘ is provided. Any value.

clipmin — Clip autorange minimum if it goes beyond this value. Has no effect when ‘autorangeoptions.minallowed‘ is provided. Any value.

include — Extends the autorange to include this value or array of values, even when they fall outside the data range. Has no effect on endpoints set by ‘autorangeoptions.minallowed‘ or ‘autorangeoptions.maxallowed‘. Subject to ‘autorangeoptions.clipmin‘ and ‘autorangeoptions.clipmax‘. Any value.

bottom — Sets the space in pixels to reserve inside the bottom edge of ‘domain‘. Applies to y axes only. Number in the interval [0, Inf].

`left` Sets the space in pixels to reserve inside the left edge of ‘domain‘. Applies to x axes only. Number in the interval [0, Inf].

`right` Sets the space in pixels to reserve inside the right edge of ‘domain‘. Applies to x axes only. Number in the interval [0, Inf].

`top` Sets the space in pixels to reserve inside the top edge of ‘domain‘. Applies to y axes only. Number in the interval [0, Inf].

`bounds` Sets the lower and upper bounds of this axis rangebreak. Can be used with ‘pattern‘. Vector or list of 2 element(s).

`dvalue` Sets the size of each ‘values‘ item. The default is one day in milliseconds. Number in the interval [0, Inf].

`enabled` `go_layout_xaxis_rangebreak()`: Determines whether this axis rangebreak is enabled or disabled. Please note that ‘rangebreaks‘ only work for "date" axis type. Logical.

`go_layout_xaxis_tickformatstop()`: Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

`name` When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`pattern` Determines a pattern on the time line that generates breaks. If "day of week" - days of the week in English e.g. ’Sunday’ or ‘sun‘ (matching is case-insensitive and considers only the first three characters), as well as Sunday-based integers between 0 and 6. If "hour" - hour (24-hour clock) as decimal numbers between 0 and 24. for more info. Examples: - { pattern: ’day of week’, bounds: [6, 1] } or simply { bounds: [’sat’, ’mon’] } breaks from Saturday to Monday (i.e. skips the weekends). - { pattern: ’hour’, bounds: [17, 8] } breaks from 5pm to 8am (i.e. skips non-work hours). One of `"day of week"`, `"hour"`, `""`.

`templateitemname` Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`values` Sets the coordinate values corresponding to the rangebreaks. An alternative to ‘bounds‘. Use ‘dvalue‘ to set the size of the values along the axis. Vector or list of up to 1 element(s).

`activecolor` Sets the background color of the active range selector button. Color: a CSS color string or R color name.

`bgcolor` `go_layout_xaxis_rangeselector()`: Sets the background color of the range selector buttons. Color: a CSS color string or R color name.

`go_layout_xaxis_rangeslider()`: Sets the background color of the range slider. Color: a CSS color string or R color name.

bordercolor
: go_layout_xaxis_rangeselector(): Sets the color of the border enclosing the range selector. Color: a CSS color string or R color name.
go_layout_xaxis_rangeslider(): Sets the border color of the range slider. Color: a CSS color string or R color name.

borderwidth
: go_layout_xaxis_rangeselector(): Sets the width (in px) of the border enclosing the range selector. Number in the interval [0, Inf].
go_layout_xaxis_rangeslider(): Sets the border width of the range slider. Integer in the interval [0, Inf].

buttons
: List of named lists or go_layout_xaxis_rangeselector_button() objects.

font
: go_layout_xaxis_rangeselector(): Sets the font of the range selector button text. Named list or go_layout_xaxis_rangeselector_font() object.
go_layout_xaxis_title(): Sets this axis' title font. Named list or go_layout_xaxis_title_font() object.

x
: Sets the x position (in normalized coordinates) of the range selector. Number in the interval [-2, 3].

xanchor
: Sets the range selector's horizontal position anchor. This anchor binds the 'x' position to the "left", "center" or "right" of the range selector. One of "auto", "left", "center", "right".

y
: Sets the y position (in normalized coordinates) of the range selector. Number in the interval [-2, 3].

yanchor
: Sets the range selector's vertical position anchor This anchor binds the 'y' position to the "top", "middle" or "bottom" of the range selector. One of "auto", "top", "middle", "bottom".

count
: Sets the number of steps to take to update the range. Use with 'step' to specify the update interval. Number in the interval [0, Inf].

label
: Sets the text label to appear on the button. String.

step
: The unit of measurement that the 'count' value will set the range by. One of "month", "year", "day", "hour", "minute", "second", "all".

stepmode
: Sets the range update mode. If "backward", the range update shifts the start of range back "count" times "step" milliseconds. If "todate", the range update shifts the start of range back to the first timestamp from "count" times "step" milliseconds back. For example, with 'step' set to "year" and 'count' set to 1 the range update shifts the start of the range back to January 01 of the current year. Month and year "todate" are currently available only for the built-in (Gregorian) calendar. One of "backward", "todate".

family
: HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren't available. String.

lineposition
: Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of "under", "over", "through" joined with "+", or one of "none".

shadow
: Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.

size	Number in the interval [1, Inf].
style	Sets whether a font should be styled with a normal or italic face from its family. One of "normal", "italic".
textcase	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of "normal", "word caps", "upper", "lower".
variant	Sets the variant of the font. One of "normal", "small-caps", "all-small-caps", "all-petite-caps", "petite-caps", "unicase".
weight	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
thickness	The height of the range slider as a fraction of the total plot area height. Number in the interval [0, 1].
yaxis	Named list or go_layout_xaxis_rangeslider_yaxis() object.
dtickrange	range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).
value	string - dtickformat for described zoom level, the same as "tickformat" String.
standoff	Sets the standoff distance (in px) between the axis labels and the title text The default value is a function of the axis tick labels, the title 'font.size' and the axis 'linewidth'. Note that the axis title position is always constrained within the margins, so the actual standoff distance is always less than the set or default value. By setting 'standoff' and turning on 'automargin', plotly.js will push the margins to fit the axis title at given standoff distance. Number in the interval [0, Inf].
text	go_layout_xaxis_title(): Sets the title of this axis. String. go_layout_xaxis_unifiedhovertitle(): Template string used for rendering the title that appear on x or y unified hover box. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. String.

Details

go_layout_xaxis() and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see plotly_obj-modify()).

Nested objects documented here:

go_layout_xaxis_autorangeoptions() layout.xaxis.autorangeoptions

`go_layout_xaxis_domainpad()` `layout.xaxis.domainpad`: Sets the space in pixels to reserve inside the edges of 'domain'. Because 'domain' is a fraction of the plot area, the gap between subplots changes with the figure size. 'domainpad' keeps the same size in pixels at all figure sizes, so it can hold content with a fixed size, such as a subplot title. x axes use 'left' and 'right'. y axes use 'top' and 'bottom'. If the padding needs more room than the domain has, the two pads shrink by the same factor, and the plot area keeps a length of at least 2 pixels. An axis that sets 'overlaying' ignores its own 'domainpad' and uses the 'domainpad' of the axis it overlays, so that both axes share the same plot area.

`go_layout_xaxis_minor()` `layout.xaxis.minor`

`go_layout_xaxis_rangebreak()` `layout.xaxis.rangebreak`

`go_layout_xaxis_rangeselector()` `layout.xaxis.rangeselector`

`go_layout_xaxis_rangeselector_button()` `layout.xaxis.rangeselector.button`: Sets the specifications for each buttons. By default, a range selector comes with no buttons.

`go_layout_xaxis_rangeselector_font()` `layout.xaxis.rangeselector.font`: Sets the font of the range selector button text.

`go_layout_xaxis_rangeslider()` `layout.xaxis.rangeslider`

`go_layout_xaxis_rangeslider_yaxis()` `layout.xaxis.rangeslider.yaxis`

`go_layout_xaxis_tickfont()` `layout.xaxis.tickfont`: Sets the tick font.

`go_layout_xaxis_tickformatstop()` `layout.xaxis.tickformatstop`

`go_layout_xaxis_title()` `layout.xaxis.title`

`go_layout_xaxis_title_font()` `layout.xaxis.title.font`: Sets this axis' title font.

`go_layout_xaxis_unifiedhovertitle()` `layout.xaxis.unifiedhovertitle`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

---

go_layout_yaxis *Layout yaxis objects*

Description

Properties of `layout.yaxis`.

Usage

```
go_layout_yaxis(
  arg = NULL,
  ...,
  anchor = NULL,
  automargin = NULL,
  autorange = NULL,
  autorangeoptions = NULL,
  autoshift = NULL,
  autotickangles = NULL,
  autotypenumbers = NULL,
  calendar = NULL,
  categoryarray = NULL,
  categoryorder = NULL,
  color = NULL,
  constrain = NULL,
  constraintoward = NULL,
  dividercolor = NULL,
  dividerwidth = NULL,
  domain = NULL,
  domainpad = NULL,
  dtick = NULL,
  exponentformat = NULL,
  fixedrange = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
  hoverformat = NULL,
  insiderange = NULL,
  labelalias = NULL,
  layer = NULL,
  linecolor = NULL,
  linewidth = NULL,
  matches = NULL,
  maxallowed = NULL,
  minallowed = NULL,
  minexponent = NULL,
  minor = NULL,
  minorloglabels = NULL,
  mirror = NULL,
  modebardisable = NULL,
  nticks = NULL,
  overlaying = NULL,
  position = NULL,
  range = NULL,
  rangebreaks = NULL,
  rangemode = NULL,
  scaleanchor = NULL,
```

```
    scaleratio = NULL,
    separatethousands = NULL,
    shift = NULL,
    showdividers = NULL,
    showexponent = NULL,
    showgrid = NULL,
    showline = NULL,
    showspikes = NULL,
    showticklabels = NULL,
    showtickprefix = NULL,
    showticksuffix = NULL,
    side = NULL,
    spikecolor = NULL,
    spikedash = NULL,
    spikemode = NULL,
    spikesnap = NULL,
    spikethickness = NULL,
    tick0 = NULL,
    tickangle = NULL,
    tickcolor = NULL,
    tickfont = NULL,
    tickformat = NULL,
    tickformatstops = NULL,
    ticklabelindex = NULL,
    ticklabelmode = NULL,
    ticklabeloverflow = NULL,
    ticklabelposition = NULL,
    ticklabelshift = NULL,
    ticklabelstandoff = NULL,
    ticklabelstep = NULL,
    ticklen = NULL,
    tickmode = NULL,
    tickprefix = NULL,
    ticks = NULL,
    tickson = NULL,
    ticksuffix = NULL,
    ticktext = NULL,
    tickvals = NULL,
    tickwidth = NULL,
    title = NULL,
    type = NULL,
    uirevision = NULL,
    unifiedhovertitle = NULL,
    visible = NULL,
    zeroline = NULL,
    zerolinecolor = NULL,
    zerolinelayer = NULL,
    zerolinewidth = NULL,
```

```
  skip_invalid = FALSE
)

go_layout_yaxis_autorangeoptions(
  arg = NULL,
  ...,
  clipmax = NULL,
  clipmin = NULL,
  include = NULL,
  maxallowed = NULL,
  minallowed = NULL,
  skip_invalid = FALSE
)

go_layout_yaxis_domainpad(
  arg = NULL,
  ...,
  bottom = NULL,
  left = NULL,
  right = NULL,
  top = NULL,
  skip_invalid = FALSE
)

go_layout_yaxis_minor(
  arg = NULL,
  ...,
  dtick = NULL,
  gridcolor = NULL,
  griddash = NULL,
  gridwidth = NULL,
  nticks = NULL,
  showgrid = NULL,
  tick0 = NULL,
  tickcolor = NULL,
  ticklen = NULL,
  tickmode = NULL,
  ticks = NULL,
  tickvals = NULL,
  tickwidth = NULL,
  skip_invalid = FALSE
)

go_layout_yaxis_rangebreak(
  arg = NULL,
  ...,
  bounds = NULL,
  dvalue = NULL,
```

```
  enabled = NULL,
  name = NULL,
  pattern = NULL,
  templateitemname = NULL,
  values = NULL,
  skip_invalid = FALSE
)

go_layout_yaxis_tickfont(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_yaxis_tickformatstop(
  arg = NULL,
  ...,
  dtickrange = NULL,
  enabled = NULL,
  name = NULL,
  templateitemname = NULL,
  value = NULL,
  skip_invalid = FALSE
)

go_layout_yaxis_title(
  arg = NULL,
  ...,
  font = NULL,
  standoff = NULL,
  text = NULL,
  skip_invalid = FALSE
)

go_layout_yaxis_title_font(
  arg = NULL,
  ...,
  color = NULL,
  family = NULL,
```

```
  lineposition = NULL,
  shadow = NULL,
  size = NULL,
  style = NULL,
  textcase = NULL,
  variant = NULL,
  weight = NULL,
  skip_invalid = FALSE
)

go_layout_yaxis_unifiedhovertitle(
  arg = NULL,
  ...,
  text = NULL,
  skip_invalid = FALSE
)
```

Arguments

arg	Optional named list, or graph object of the same type, to initialise properties from (constructors). Named arguments take precedence.
...	Further properties, including nested properties in "magic underscore" form, e.g. `marker_size = 10` for `marker = list(size = 10)`. Property values may be one-sided formulas, evaluated in `data` for `add_*()` functions.
anchor	If set to an opposite-letter axis id (e.g. ‘x2‘, ‘y‘), this axis is bound to the corresponding opposite-letter axis. If set to "free", this axis’ position is determined by ‘position‘. One of `"free"`, or a string matching `"^x([2-9]|[1-9][0-9]+)?( domain)?$"` or `"^y([2-9]\|[1-9][0-9]+)?( domain)?$"`.
automargin	Determines whether long tick labels automatically grow the figure margins. Any combination of `"height"`, `"width"`, `"left"`, `"right"`, `"top"`, `"bottom"` joined with `"+"`, or one of `TRUE`, `FALSE`.
autorange	Determines whether or not the range of this axis is computed in relation to the input data. See ‘rangemode‘ for more info. If ‘range‘ is provided and it has a value for both the lower and upper bound, ‘autorange‘ is set to FALSE. Using "min" applies autorange only to set the minimum. Using "max" applies autorange only to set the maximum. Using "min reversed" applies autorange only to set the minimum on a reversed axis. Using "max reversed" applies autorange only to set the maximum on a reversed axis. Using "reversed" applies autorange on both ends and reverses the axis direction. One of `"reversed"`, `"min reversed"`, `"max reversed"`, `"min"`, `"max"`, `TRUE`, `FALSE`.
autorangeoptions	Named list or `go_layout_yaxis_autorangeoptions()` object.
autoshift	Automatically reposition the axis to avoid overlap with other axes with the same ‘overlaying‘ value. This repositioning will account for any ‘shift‘ amount applied to other axes on the same side with ‘autoshift‘ is set to true. Only has an effect if ‘anchor‘ is set to "free". Logical.

autotickangles When ‘tickangle‘ is set to "auto", it will be set to the first angle in this array that is large enough to prevent label overlap. Vector or list of up to 1 element(s).

autotypenumbers
Using "strict" a numeric string in trace data is not converted to a number. Using "convert types" a numeric string in trace data may be treated as a number during automatic axis ‘type‘ detection. Defaults to layout.autotypenumbers. One of "convert types", "strict".

calendar Sets the calendar system to use for ‘range‘ and ‘tick0‘ if this is a date axis. This does not set the calendar for interpreting data on this axis, that's specified in the trace or via the global ‘layout.calendar‘ One of "chinese", "coptic", "discworld", "ethiopian", "gregorian", "hebrew", "islamic", "jalali", "julian", "mayan", "nanakshahi", "nepali", "persian", "taiwan", "thai", "ummalqura".

categoryarray Sets the order in which categories on this axis appear. Only has an effect if ‘categoryorder‘ is set to "array". Used with ‘categoryorder‘. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

categoryorder Specifies the ordering logic for the case of categorical variables. By default, plotly uses "trace", which specifies the order that is present in the data supplied. Set ‘categoryorder‘ to "category ascending" or "category descending" if order should be determined by the alphanumerical order of the category names. Set ‘categoryorder‘ to "array" to derive the ordering from the attribute ‘categoryarray‘. If a category is not found in the ‘categoryarray‘ array, the sorting behavior for that attribute will be identical to the "trace" mode. The unspecified categories will follow the categories in ‘categoryarray‘. Set ‘categoryorder‘ to "total ascending" or "total descending" if order should be determined by the numerical order of the values. Similarly, the order can be determined by the min, max, sum, mean, geometric mean or median of all the values. One of "trace", "category ascending", "category descending", "array", "total ascending", "total descending", "min ascending", "min descending", "max ascending", "max descending", "sum ascending", "sum descending", "mean ascending", "mean descending", "geometric mean ascending", "geometric mean descending", "median ascending", "median descending".

color go_layout_yaxis(): Sets default for all colors associated with this axis all at once: line, font, tick, and grid colors. Grid color is lightened by blending this with the plot background Individual pieces can override this. Color: a CSS color string or R color name.

go_layout_yaxis_tickfont(), go_layout_yaxis_title_font(): Color: a CSS color string or R color name.

constrain If this axis needs to be compressed (either due to its own ‘scaleanchor‘ and ‘scaleratio‘ or those of the other axis), determines how that happens: by increasing the "range", or by decreasing the "domain". Default is "domain" for axes containing image traces, "range" otherwise. One of "range", "domain".

constraintoward
If this axis needs to be compressed (either due to its own ‘scaleanchor‘ and ‘scaleratio‘ or those of the other axis), determines which direction we push the originally specified plot area. Options are "left", "center" (default), and "right"

for x axes, and "top", "middle" (default), and "bottom" for y axes. One of `"left"`, `"center"`, `"right"`, `"top"`, `"middle"`, `"bottom"`.

`dividercolor` Sets the color of the dividers Only has an effect on "multicategory" axes. Color: a CSS color string or R color name.

`dividerwidth` Sets the width (in px) of the dividers Only has an effect on "multicategory" axes. Number.

`domain` Sets the domain of this axis (in plot fraction). Vector or list of 2 element(s).

`domainpad` Sets the space in pixels to reserve inside the edges of 'domain'. Because 'domain' is a fraction of the plot area, the gap between subplots changes with the figure size. 'domainpad' keeps the same size in pixels at all figure sizes, so it can hold content with a fixed size, such as a subplot title. x axes use 'left' and 'right'. y axes use 'top' and 'bottom'. If the padding needs more room than the domain has, the two pads shrink by the same factor, and the plot area keeps a length of at least 2 pixels. An axis that sets 'overlaying' ignores its own 'domainpad' and uses the 'domainpad' of the axis it overlays, so that both axes share the same plot area. Named list or `go_layout_yaxis_domainpad()` object.

`dtick` Sets the step in-between ticks on this axis. Use with 'tick0'. Must be a positive number, or special strings available to "log" and "date" axes. If the axis 'type' is "log", then ticks are set every 10^(n*dtick) where n is the tick number. For example, to set a tick mark at 1, 10, 100, 1000, ... set dtick to 1. To set tick marks at 1, 100, 10000, ... set dtick to 2. To set tick marks at 1, 5, 25, 125, 625, 3125, ... set dtick to log_10(5), or 0.69897000433. "log" has several special values; "L<f>", where 'f' is a positive number, gives ticks linearly spaced in value (but not position). For example 'tick0' = 0.1, 'dtick' = "L0.5" will put ticks at 0.1, 0.6, 1.1, 1.6 etc. To show powers of 10 plus small digits between, use "D1" (all digits) or "D2" (only 2 and 5). 'tick0' is ignored for "D1" and "D2". If the axis 'type' is "date", then you must convert the time to milliseconds. For example, to set the interval between ticks to one day, set 'dtick' to 86400000.0. "date" also has special values "M<n>" gives ticks spaced by a number of months. 'n' must be a positive integer. To set ticks on the 15th of every third month, set 'tick0' to "2000-01-15" and 'dtick' to "M3". To set ticks every 4 years, set 'dtick' to "M48" Any value.

`exponentformat` Determines a formatting rule for the tick exponents. For example, consider the number 1,000,000,000. If "none", it appears as 1,000,000,000. If "e", 1e+9. If "E", 1E+9. If "power", 1x10^9 (with 9 in a super script). If "SI", 1G. If "B", 1B. "SI" uses prefixes from "femto" f (10^-15) to "tera" T (10^12). "SI extended" covers instead the full SI range from "quecto" q (10^-30) to "quetta" Q (10^30). If "SI" or "SI extended" is used and the exponent is beyond the above ranges, the formatting rule will automatically be switched to the power notation. One of `"none"`, `"e"`, `"E"`, `"power"`, `"SI"`, `"B"`, `"SI extended"`.

`fixedrange` Determines whether or not this axis is zoom-able. If true, then zoom is disabled. Logical.

`gridcolor` Sets the color of the grid lines. Color: a CSS color string or R color name.

`griddash` Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of `"solid"`, `"dot"`, `"dash"`, `"longdash"`, `"dashdot"`, `"longdashdot"`, or a string matching `"^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$"`.

`gridwidth` Sets the width (in px) of the grid lines. Number in the interval [0, Inf].

`hoverformat` Sets the hover text formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

`insiderange` Could be used to set the desired inside range of this axis (excluding the labels) when 'ticklabelposition' of the anchored axis has "inside". Not implemented for axes with 'type' "log". This would be ignored when 'range' is provided. Vector or list of 2 element(s).

`labelalias` Replacement text for specific tick or hover labels. For example using {US: 'USA', CA: 'Canada'} changes US to USA and CA to Canada. The labels we would have shown must match the keys exactly, after adding any tickprefix or ticksuffix. For negative numbers the minus sign symbol used (U+2212) is wider than the regular ascii dash. That means you need to use -1 instead of -1. labelalias can be used with any axis type, and both keys (if needed) and values (if desired) can include html-like tags or MathJax. Any value.

`layer` Sets the layer on which this axis is displayed. If "above traces", this axis is displayed above all the subplot's traces If "below traces", this axis is displayed below all the subplot's traces, but above the grid lines. Useful when used together with scatter-like traces with 'cliponaxis' set to FALSE to show markers and/or text nodes above this axis. One of `"above traces"`, `"below traces"`.

`linecolor` Sets the axis line color. Color: a CSS color string or R color name.

`linewidth` Sets the width (in px) of the axis line. Number in the interval [0, Inf].

`matches` If set to another axis id (e.g. 'x2', 'y'), the range of this axis will match the range of the corresponding axis in data-coordinates space. Moreover, matching axes share auto-range values, category lists and histogram auto-bins. Note that setting axes simultaneously in both a 'scaleanchor' and a 'matches' constraint is currently forbidden. Moreover, note that matching axes must have the same 'type'. One of , or a string matching `"^x([2-9]|[1-9][0-9]+)?( domain)?$"` or `"^y([2-9]|[1-9][0-9]+)?( domain)?$"`.

`maxallowed` `go_layout_yaxis()`: Determines the maximum range of this axis. Any value.

`go_layout_yaxis_autorangeoptions()`: Use this value exactly as autorange maximum. Any value.

`minallowed` `go_layout_yaxis()`: Determines the minimum range of this axis. Any value.

`go_layout_yaxis_autorangeoptions()`: Use this value exactly as autorange minimum. Any value.

`minexponent` Hide SI prefix for 10^n if |n| is below this number. This only has an effect when 'tickformat' is "SI" or "B". Number in the interval [0, Inf].

`minor` Named list or `go_layout_yaxis_minor()` object.

`minorloglabels` Determines how minor log labels are displayed. If "small digits", small digits i.e. 2 or 5 are displayed. If "complete", complete digits are displayed. If "none", no labels are displayed. One of `"small digits"`, `"complete"`, `"none"`.

`mirror` Determines if the axis lines or/and ticks are mirrored to the opposite side of the plotting area. If TRUE, the axis lines are mirrored. If "ticks", the axis lines and ticks are mirrored. If FALSE, mirroring is disable. If "all", axis lines are mirrored on all shared-axes subplots. If "allticks", axis lines and ticks are mirrored on all shared-axes subplots. One of `"ticks"`, `"all"`, `"allticks"`, `TRUE`, `FALSE`.

`modebardisable` Disables certain modebar buttons for this axis. "autoscale" disables the autoscale buttons, "zoominout" disables the zoom-in and zoom-out buttons. Any combination of `"autoscale"`, `"zoominout"` joined with `"+"`, or one of `"none"`.

`nticks` Specifies the maximum number of ticks for the particular axis. The actual number of ticks will be chosen automatically to be less than or equal to ‘nticks‘. Has an effect only if ‘tickmode‘ is set to "auto". Integer in the interval [0, Inf].

`overlaying` If set a same-letter axis id, this axis is overlaid on top of the corresponding same-letter axis, with traces and axes visible for both axes. If FALSE, this axis does not overlay any same-letter axes. In this case, for axes with overlapping domains only the highest-numbered axis will be visible. One of `"free"`, or a string matching `"^x([2-9]|[1-9][0-9]+)?( domain)?$"` or `"^y([2-9]|[1-9][0-9]+)?( domain)?$"`.

`position` Sets the position of this axis in the plotting space (in normalized coordinates). Only has an effect if ‘anchor‘ is set to "free". Number in the interval [0, 1].

`range` Sets the range of this axis. If the axis ‘type‘ is "log", then you must take the log of your desired range (e.g. to set the range from 1 to 100, set the range from 0 to 2). If the axis ‘type‘ is "date", it should be date strings, like date data, though Date objects and unix milliseconds will be accepted and converted to strings. If the axis ‘type‘ is "category", it should be numbers, using the scale where each category is assigned a serial number from zero in the order it appears. Leaving either or both elements ‘null‘ impacts the default ‘autorange‘. Vector or list of 2 element(s).

`rangebreaks` List of named lists or `go_layout_yaxis_rangebreak()` objects.

`rangemode` If "normal", the range is computed in relation to the extrema of the input data. If "tozero", the range extends to 0, regardless of the input data If "nonnegative", the range is non-negative, regardless of the input data. Applies only to linear axes. One of `"normal"`, `"tozero"`, `"nonnegative"`.

`scaleanchor` If set to another axis id (e.g. ‘x2‘, ‘y‘), the range of this axis changes together with the range of the corresponding axis such that the scale of pixels per unit is in a constant ratio. Both axes are still zoomable, but when you zoom one, the other will zoom the same amount, keeping a fixed midpoint. ‘constrain‘ and ‘constraintoward‘ determine how we enforce the constraint. You can chain these, ie ‘yaxis: {scaleanchor: "x"}, xaxis2: {scaleanchor: "y"}‘ but you can only link axes of the same ‘type‘. The linked axis can have the opposite letter (to constrain the aspect ratio) or the same letter (to match scales across subplots). Loops (‘yaxis: {scaleanchor: "x"}, xaxis: {scaleanchor: "y"}‘ or longer) are redundant and the last constraint encountered will be ignored to avoid possible inconsistent constraints via ‘scaleratio‘. Note that setting axes simultaneously in both a ‘scaleanchor‘ and a ‘matches‘ constraint is currently forbidden. Setting ‘false‘ allows to remove a default constraint (occasionally, you may need to prevent a

default ‘scaleanchor‘ constraint from being applied, eg. when having an image trace ‘yaxis: {scaleanchor: "x"}‘ is set automatically in order for pixels to be rendered as squares, setting ‘yaxis: {scaleanchor: false}‘ allows to remove the constraint). One of FALSE, or a string matching "^x([2-9]|[1-9][0-9]+)?(domain)?$" or "^y([2-9]|[1-9][0-9]+)?(domain)?$".

scaleratio If this axis is linked to another by ‘scaleanchor‘, this determines the pixel to unit scale ratio. For example, if this value is 10, then every unit on this axis spans 10 times the number of pixels as a unit on the linked axis. Use this for example to create an elevation profile where the vertical scale is exaggerated a fixed amount with respect to the horizontal. Number in the interval [0, Inf].

separatethousands
If "true", even 4-digit integers are separated Logical.

shift Moves the axis a given number of pixels from where it would have been otherwise. Accepts both positive and negative values, which will shift the axis either right or left, respectively. If ‘autoshift‘ is set to true, then this defaults to a padding of -3 if ‘side‘ is set to "left". and defaults to +3 if ‘side‘ is set to "right". Defaults to 0 if ‘autoshift‘ is set to false. Only has an effect if ‘anchor‘ is set to "free". Number.

showdividers Determines whether or not a dividers are drawn between the category levels of this axis. Only has an effect on "multicategory" axes. Logical.

showexponent If "all", all exponents are shown besides their significands. If "first", only the exponent of the first tick is shown. If "last", only the exponent of the last tick is shown. If "none", no exponents appear. One of "all", "first", "last", "none".

showgrid Determines whether or not grid lines are drawn. If TRUE, the grid lines are drawn at every tick mark. Logical.

showline Determines whether or not a line bounding this axis is drawn. Logical.

showspikes Determines whether or not spikes (aka droplines) are drawn for this axis. Note that spikes will never be drawn when ‘hovermode‘ is FALSE. Logical.

showticklabels Determines whether or not the tick labels are drawn. Logical.

showtickprefix If "all", all tick labels are displayed with a prefix. If "first", only the first tick is displayed with a prefix. If "last", only the last tick is displayed with a suffix. If "none", tick prefixes are hidden. One of "all", "first", "last", "none".

showticksuffix Same as ‘showtickprefix‘ but for tick suffixes. One of "all", "first", "last", "none".

side Determines whether a x (y) axis is positioned at the "bottom" ("left") or "top" ("right") of the plotting area. One of "top", "bottom", "left", "right".

spikecolor Sets the spike color. If undefined, will use the series color Color: a CSS color string or R color name.

spikedash Sets the dash style of lines. Set to a dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px"). One of "solid", "dot", "dash", "longdash", "dashdot", "longdashdot", or a string matching "^\\d+(\\.\\d+)?(px|%)?((,|\\s)\\s*\\d+(\\.\\d+)?(px|%)?)*$".

spikemode Determines the drawing mode for the spike line If "toaxis", the line is drawn from the data point to the axis the series is plotted on. If "across", the line is drawn across the entire plot area, and supercedes "toaxis". If "marker", then a marker dot is drawn on the axis the series is plotted on Any combination of "toaxis", "across", "marker" joined with "+".

spikesnap Determines whether spikelines are stuck to the cursor or to the closest datapoints. One of "data", "cursor", "hovered data".

spikethickness Sets the spike line width in pixels. Number.

tick0 Sets the placement of the first tick on this axis. Use with 'dtick'. If the axis 'type' is "log", then you must take the log of your starting tick (e.g. to set the starting tick to 100, set the 'tick0' to 2) except when 'dtick'="L<f>" (see 'dtick' for more info). If the axis 'type' is "date", it should be a date string, like date data. If the axis 'type' is "category", it should be a number, using the scale where each category is assigned a serial number from zero in the order it appears. Any value.

tickangle Sets the angle of the tick labels with respect to the horizontal. For example, a 'tickangle' of -90 draws the tick labels vertically. Angle in degrees.

tickcolor Sets the tick color. Color: a CSS color string or R color name.

tickfont Sets the tick font. Named list or go_layout_yaxis_tickfont() object.

tickformat Sets the tick label formatting rule using d3 formatting mini-languages which are very similar to those in Python. For numbers, see: https://github.com/d3/d3-format/tree/v1.4.5#d3-format. And for dates see: https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format. We add two items to d3's date formatter: "%h" for half of the year as a decimal number as well as "%{n}f" for fractional seconds with n digits. For example, "2016-10-13 09:15:23.456" with tickformat "%H~%M~%S.%2f" would display "09~15~23.46" String.

tickformatstops List of named lists or go_layout_yaxis_tickformatstop() objects.

ticklabelindex Only for axes with 'type' "date" or "linear". Instead of drawing the major tick label, draw the label for the minor tick that is n positions away from the major tick. E.g. to always draw the label for the minor tick before each major tick, choose 'ticklabelindex' -1. This is useful for date axes with 'ticklabelmode' "period" if you want to label the period that ends with each major tick instead of the period that begins there. Integer, or a vector of these.

ticklabelmode Determines where tick labels are drawn with respect to their corresponding ticks and grid lines. Only has an effect for axes of 'type' "date" When set to "period", tick labels are drawn in the middle of the period between ticks. One of "instant", "period".

ticklabeloverflow Determines how we handle tick labels that would overflow either the graph div or the domain of the axis. The default value for inside tick labels is "hide past domain". Otherwise on "category" and "multicategory" axes the default is "allow". In other cases the default is "hide past div". One of "allow", "hide past div", "hide past domain".

`ticklabelposition`
: Determines where tick labels are drawn with respect to the axis. Please note that top or bottom has no effect on x axes or when ‘ticklabelmode‘ is set to "period" or when ‘tickson‘ is set to "boundaries". Similarly, left or right has no effect on y axes or when ‘ticklabelmode‘ is set to "period" or when ‘tickson‘ is set to "boundaries". Has no effect on "multicategory" axes. When used on axes linked by ‘matches‘ or ‘scaleanchor‘, no extra padding for inside labels would be added by autorange, so that the scales could match. One of `"outside"`, `"inside"`, `"outside top"`, `"inside top"`, `"outside left"`, `"inside left"`, `"outside right"`, `"inside right"`, `"outside bottom"`, `"inside bottom"`.

`ticklabelshift`
: Shifts the tick labels by the specified number of pixels in parallel to the axis. Positive values move the labels in the positive direction of the axis. Integer.

`ticklabelstandoff`
: Sets the standoff distance (in px) between the axis tick labels and their default position. A positive ‘ticklabelstandoff‘ moves the labels farther away from the plot area if ‘ticklabelposition‘ is "outside", and deeper into the plot area if ‘ticklabelposition‘ is "inside". A negative ‘ticklabelstandoff‘ works in the opposite direction, moving outside ticks towards the plot area and inside ticks towards the outside. If the negative value is large enough, inside ticks can even end up outside and vice versa. Integer.

`ticklabelstep`
: Sets the spacing between tick labels as compared to the spacing between ticks. A value of 1 (default) means each tick gets a label. A value of 2 means shows every 2nd label. A larger value n means only every nth tick is labeled. ‘tick0‘ determines which labels are shown. Not implemented for axes with ‘type‘ "log" or "multicategory", or when ‘tickmode‘ is "array". Integer in the interval [1, Inf].

`ticklen`
: Sets the tick length (in px). Number in the interval [0, Inf].

`tickmode`
: `go_layout_yaxis()`: Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). If "sync", the number of ticks will sync with the overlayed axis set by ‘overlaying‘ property. When no other tick info is provided, overlaying (non-categorical) axes default to "sync", while other axes default to "auto". One of `"auto"`, `"linear"`, `"array"`, `"sync"`.

  `go_layout_yaxis_minor()`: Sets the tick mode for this axis. If "auto", the number of ticks is set via ‘nticks‘. If "linear", the placement of the ticks is determined by a starting position ‘tick0‘ and a tick step ‘dtick‘ ("linear" is the default value if ‘tick0‘ and ‘dtick‘ are provided). If "array", the placement of the ticks is set via ‘tickvals‘ and the tick text is ‘ticktext‘. ("array" is the default value if ‘tickvals‘ is provided). One of `"auto"`, `"linear"`, `"array"`.

`tickprefix`
: Sets a tick label prefix. String.

`ticks`
: Determines whether ticks are drawn or not. If **, this axis’ ticks are not drawn. If "outside" ("inside"), this axis’ are drawn outside (inside) the axis lines. One of `"outside"`, `"inside"`, `""`.

`tickson` Determines where ticks and grid lines are drawn with respect to their corresponding tick labels. Only has an effect for axes of 'type' "category" or "multicategory". When set to "boundaries", ticks and grid lines are drawn half a category to the left/bottom of labels. One of `"labels"`, `"boundaries"`.

`ticksuffix` Sets a tick label suffix. String.

`ticktext` Sets the text displayed at the ticks position via 'tickvals'. Only has an effect if 'tickmode' is set to "array". Used with 'tickvals'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickvals` Sets the values at which ticks on this axis appear. Only has an effect if 'tickmode' is set to "array". Used with 'ticktext'. Data array: a vector, factor, Date/POSIXct vector, matrix or list.

`tickwidth` Sets the tick width (in px). Number in the interval [0, Inf].

`title` String (the title text), named list or `go_layout_yaxis_title()` object.

`type` Sets the axis type. By default, plotly attempts to determined the axis type by looking into the data of the traces that referenced the axis in question. One of `"-"`, `"linear"`, `"log"`, `"date"`, `"category"`, `"multicategory"`.

`uirevision` Controls persistence of user-driven changes in axis 'range', 'autorange', and 'title' if in 'editable: true' configuration. Defaults to 'layout.uirevision'. Any value.

`unifiedhovertitle`
Named list or `go_layout_yaxis_unifiedhovertitle()` object.

`visible` A single toggle to hide the axis while preserving interaction like dragging. Default is true when a cheater plot is present on the axis, otherwise false Logical.

`zeroline` Determines whether or not a line is drawn at along the 0 value of this axis. If TRUE, the zero line is drawn on top of the grid lines. Logical.

`zerolinecolor` Sets the line color of the zero line. Color: a CSS color string or R color name.

`zerolinelayer` Sets the layer on which this zeroline is displayed. If "above traces", this zeroline is displayed above all the subplot's traces If "below traces", this zeroline is displayed below all the subplot's traces, but above the grid lines. Limitation: "zerolinelayer" currently has no effect if the "zorder" property is set on any trace. One of `"above traces"`, `"below traces"`.

`zerolinewidth` Sets the width (in px) of the zero line. Number.

`skip_invalid` If `TRUE`, silently drop invalid properties instead of raising an error.

`clipmax` Clip autorange maximum if it goes beyond this value. Has no effect when 'autorangeoptions.maxallowed' is provided. Any value.

`clipmin` Clip autorange minimum if it goes beyond this value. Has no effect when 'autorangeoptions.minallowed' is provided. Any value.

`include` Extends the autorange to include this value or array of values, even when they fall outside the data range. Has no effect on endpoints set by 'autorangeoptions.minallowed' or 'autorangeoptions.maxallowed'. Subject to 'autorangeoptions.clipmin' and 'autorangeoptions.clipmax'. Any value.

`bottom` Sets the space in pixels to reserve inside the bottom edge of 'domain'. Applies to y axes only. Number in the interval [0, Inf].

`left` Sets the space in pixels to reserve inside the left edge of ‘domain‘. Applies to x axes only. Number in the interval [0, Inf].

`right` Sets the space in pixels to reserve inside the right edge of ‘domain‘. Applies to x axes only. Number in the interval [0, Inf].

`top` Sets the space in pixels to reserve inside the top edge of ‘domain‘. Applies to y axes only. Number in the interval [0, Inf].

`bounds` Sets the lower and upper bounds of this axis rangebreak. Can be used with ‘pattern‘. Vector or list of 2 element(s).

`dvalue` Sets the size of each ‘values‘ item. The default is one day in milliseconds. Number in the interval [0, Inf].

`enabled` `go_layout_yaxis_rangebreak()`: Determines whether this axis rangebreak is enabled or disabled. Please note that ‘rangebreaks‘ only work for "date" axis type. Logical.

`go_layout_yaxis_tickformatstop()`: Determines whether or not this stop is used. If ‘false‘, this stop is ignored even within its ‘dtickrange‘. Logical.

`name` When used in a template, named items are created in the output figure in addition to any items the figure already has in this array. You can modify these items in the output figure by making your own item with ‘templateitemname‘ matching this ‘name‘ alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). Has no effect outside of a template. String.

`pattern` Determines a pattern on the time line that generates breaks. If "day of week" - days of the week in English e.g. ’Sunday’ or ‘sun‘ (matching is case-insensitive and considers only the first three characters), as well as Sunday-based integers between 0 and 6. If "hour" - hour (24-hour clock) as decimal numbers between 0 and 24. for more info. Examples: - { pattern: ’day of week’, bounds: [6, 1] } or simply { bounds: [’sat’, ’mon’] } breaks from Saturday to Monday (i.e. skips the weekends). - { pattern: ’hour’, bounds: [17, 8] } breaks from 5pm to 8am (i.e. skips non-work hours). One of `"day of week"`, `"hour"`, `""`.

`templateitemname`
Used to refer to a named item in this array in the template. Named items from the template will be created even without a matching item in the input figure, but you can modify one by making an item with ‘templateitemname‘ matching its ‘name‘, alongside your modifications (including ‘visible: false‘ or ‘enabled: false‘ to hide it). If there is no template or no matching item, this item will be hidden unless you explicitly show it with ‘visible: true‘. String.

`values` Sets the coordinate values corresponding to the rangebreaks. An alternative to ‘bounds‘. Use ‘dvalue‘ to set the size of the values along the axis. Vector or list of up to 1 element(s).

`family` HTML font family - the typeface that will be applied by the web browser. The web browser can only apply a font if it is available on the system where it runs. Provide multiple font families, separated by commas, to indicate the order in which to apply fonts if they aren’t available. String.

`lineposition` Sets the kind of decoration line(s) with text, such as an "under", "over" or "through" as well as combinations e.g. "under+over", etc. Any combination of `"under"`, `"over"`, `"through"` joined with `"+"`, or one of `"none"`.

shadow	Sets the shape and color of the shadow behind text. "auto" places minimal shadow and applies contrast text font color. See https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow for additional options. String.
size	Number in the interval [1, Inf].
style	Sets whether a font should be styled with a normal or italic face from its family. One of `"normal"`, `"italic"`.
textcase	Sets capitalization of text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized. One of `"normal"`, `"word caps"`, `"upper"`, `"lower"`.
variant	Sets the variant of the font. One of `"normal"`, `"small-caps"`, `"all-small-caps"`, `"all-petite-caps"`, `"petite-caps"`, `"unicase"`.
weight	Sets the weight (or boldness) of the font. Integer in the interval [1, 1000].
dtickrange	range ["min", "max"], where "min", "max" - dtick values which describe some zoom level, it is possible to omit "min" or "max" value by passing "null" Vector or list of 2 element(s).
value	string - dtickformat for described zoom level, the same as "tickformat" String.
font	Sets this axis' title font. Named list or `go_layout_yaxis_title_font()` object.
standoff	Sets the standoff distance (in px) between the axis labels and the title text The default value is a function of the axis tick labels, the title 'font.size' and the axis 'linewidth'. Note that the axis title position is always constrained within the margins, so the actual standoff distance is always less than the set or default value. By setting 'standoff' and turning on 'automargin', plotly.js will push the margins to fit the axis title at given standoff distance. Number in the interval [0, Inf].
text	`go_layout_yaxis_title()`: Sets the title of this axis. String. `go_layout_yaxis_unifiedhovertitle()`: Template string used for rendering the title that appear on x or y unified hover box. Variables are inserted using %{variable}, for example "y: %{y}". Numbers are formatted using d3-format's syntax %{variable:d3-format}, for example "Price: %{y:$.2f}". https://github.com/d3/d3-format/tree/v1.4.5#d3-format for details on the formatting syntax. Dates are formatted using d3-time-format's syntax %{variable|d3-time-format}, for example "Day: %{2019-01-01|%A}". https://github.com/d3/d3-time-format/tree/v2.2.3#locale_format for details on the date formatting syntax. Variables that can't be found will be replaced with the specifier. For example, a template of "data: %{x}, %{y}" will result in a value of "data: 1, %{y}" if x is 1 and y is missing. Variables with an undefined value will be replaced with the fallback value. String.

Details

`go_layout_yaxis()` and the nested object constructors below create validated graph objects (named lists). Every property is checked against the plotly.js schema when an object is created or modified (see `plotly_obj-modify()`).

Nested objects documented here:

`go_layout_yaxis_autorangeoptions()` `layout.yaxis.autorangeoptions`

`go_layout_yaxis_domainpad()` `layout.yaxis.domainpad`: Sets the space in pixels to reserve inside the edges of 'domain'. Because 'domain' is a fraction of the plot area, the gap between subplots changes with the figure size. 'domainpad' keeps the same size in pixels at all figure sizes, so it can hold content with a fixed size, such as a subplot title. x axes use 'left' and 'right'. y axes use 'top' and 'bottom'. If the padding needs more room than the domain has, the two pads shrink by the same factor, and the plot area keeps a length of at least 2 pixels. An axis that sets 'overlaying' ignores its own 'domainpad' and uses the 'domainpad' of the axis it overlays, so that both axes share the same plot area.

`go_layout_yaxis_minor()` `layout.yaxis.minor`

`go_layout_yaxis_rangebreak()` `layout.yaxis.rangebreak`

`go_layout_yaxis_tickfont()` `layout.yaxis.tickfont`: Sets the tick font.

`go_layout_yaxis_tickformatstop()` `layout.yaxis.tickformatstop`

`go_layout_yaxis_title()` `layout.yaxis.title`

`go_layout_yaxis_title_font()` `layout.yaxis.title.font`: Sets this axis' title font.

`go_layout_yaxis_unifiedhovertitle()` `layout.yaxis.unifiedhovertitle`

Value

Each `go_*()` function returns a graph object of the same class as its name; `add_*()` functions return the modified figure.

See Also

`go_layout()`

---

highlight *Highlight linked selections*

Description

Sets how a figure responds to, and shows, selections of points linked with crosstalk. A figure is linked when its data come from a crosstalk::SharedData (give one as the `data_frame` of a `px_*()` function, pipe one into `go_figure()` and add traces with formulas such as `add_scatter(x = ~col)`, or link traces with `set_crosstalk_keys()`): selecting points then selects the rows with the same keys in every view of the same crosstalk group, such as other plots, `crosstalk::filter_select()` inputs, DT tables or leaflet maps. Linked figures highlight on click by default; `highlight()` changes that.

Usage

```
highlight(
  p,
  on = "plotly_click",
  off,
  persistent = getOption("persistent", FALSE),
```

```
  dynamic = FALSE,
  color = NULL,
  selectize = FALSE,
  defaultValues = NULL,
  opacityDim = getOption("opacityDim", 0.2),
  selected = attrs_selected(),
  debounce = 0,
  ...
)

attrs_selected(opacity = 1, ...)
```

Arguments

p	A go_figure() (e.g. from a px_*() function) or a plotly htmlwidget.
on	The event that selects points: "plotly_click", "plotly_hover", "plotly_selected" (box or lasso selection) or "plotly_selecting" (selection while dragging). NULL turns selecting from this plot off.
off	The event that clears the selection: "plotly_doubleclick", "plotly_deselect" or "plotly_relayout". The default matches on: "plotly_deselect" for selections, otherwise "plotly_doubleclick". NULL turns clearing off.
persistent	Whether new selections add to the current one rather than replace it. Holding the shift key while selecting does this too.
dynamic	Whether to show a colour picker for the selection colour.
color	The selection colour (several, for dynamic = TRUE): R colour names, hex colours or CSS colours such as "rgba(255,0,0,0.5)". NULL keeps the points' own colours and dims the rest.
selectize	Whether to show a searchable dropdown of the keys, which selects them; or a list of selectize.js options for it.
defaultValues	Keys selected when the figure is first shown.
opacityDim	Opacity multiplier (0 to 1) of points not selected.
selected	Trace attributes of the selected points, made with attrs_selected().
debounce	Milliseconds to wait before selecting, for frequent events such as "plotly_hover" and "plotly_selecting".
...	Unused.
opacity	Opacity of the selected points.

Details

The options match the plotly package's highlight(), and highlight() also works on plotly htmlwidgets.

Value

p with the highlighting options. attrs_selected() returns a list of trace attributes.

See Also

set_crosstalk_keys()

Examples

```
sd <- crosstalk::SharedData$new(px_data_iris())
px_scatter(sd, x = ~sepal_width, y = ~sepal_length, color = ~species) |>
  highlight(on = "plotly_selected", color = "red")
```

---

install_chrome *Set up Chrome for static image export*

Description

Makes sure save_image() can start Chrome, without asking: finds a Chrome or Chromium (see save_image()) or else downloads Chrome Headless Shell with chromote, and on Linux downloads the shared libraries it needs that are missing. The libraries come from the system's own package repositories (with `apt-get download` or `dnf download`, which need no administrator rights) and are unpacked into plotly.core's user cache (`tools::R_user_dir("plotly.core", "cache")`), where `save_image()` finds them. Run it once on a new machine, e.g. before scripts that export images; interactive sessions are offered the same downloads when needed.

Usage

```
install_chrome()

remove_chrome(browser = FALSE)
```

Arguments

browser	For `remove_chrome()`, also remove the Chrome Headless Shell builds downloaded with chromote.

Details

Downloaded libraries are only used for what the system lacks: libraries installed on the system later (and its updates) take over. They are kept in a folder for the OS release, and those for other releases (e.g. after an upgrade) are removed when libraries are next downloaded. They don't get the system's updates, so once they are more than `getOption("plotly.core.chrome_libs_max_age", 90)` days old, or Chrome has changed, `install_chrome()` downloads them again (interactive sessions are asked; scripts get a message saying to run `install_chrome()`). `remove_chrome()` removes everything plotly.core has downloaded and, with `browser = TRUE`, the Chrome Headless Shell builds chromote downloaded.

The downloaded software is not part of plotly.core and keeps its own licence: Chrome Headless Shell is built from Chromium by Google (see the licence files in its download), and the libraries are your distribution's packages (each package's copyright file is kept with it, under `usr/share/doc`).

Value

`install_chrome()`: the path to Chrome, invisibly. `remove_chrome()`: `NULL`, invisibly.

Examples

```
## Not run:
install_chrome()

# remove the downloaded libraries (and Chrome Headless Shell)
remove_chrome(browser = TRUE)

## End(Not run)
```

---

last_plot *The last plot*

Description

`last_plot()` returns the plot most recently printed (or, for the plotly package's `plot_ly()` objects, built). `set_last_plot()` records one; it is for packages that build plots of their own classes.

Usage

```
last_plot()

set_last_plot(p)
```

Arguments

p	A plot.

Value

`last_plot()`: the plot. `set_last_plot()`: `p`, invisibly.

---

make_subplots *Create a figure with a grid of subplots*

Description

Lays out a `rows` x `cols` grid of subplots, ported from plotly.py's `make_subplots()`. Traces and layout objects can then be placed in a cell with the `row` and `col` arguments of `add_trace()`, the `add_<trace>()` functions, `update_xaxes()`, `add_annotation()` and similar functions.

Usage

```
make_subplots(
  rows = 1,
  cols = 1,
  shared_xaxes = FALSE,
  shared_yaxes = FALSE,
  start_cell = "top-left",
  print_grid = FALSE,
  horizontal_spacing = NULL,
  vertical_spacing = NULL,
  subplot_titles = NULL,
  column_widths = NULL,
  row_heights = NULL,
  specs = NULL,
  insets = NULL,
  column_titles = NULL,
  row_titles = NULL,
  x_title = NULL,
  y_title = NULL,
  figure = NULL,
  font = NULL
)
```

Arguments

`rows`, `cols`	Number of rows and columns in the grid.
`shared_xaxes`, `shared_yaxes`	Link axes so they pan and zoom together: `TRUE` (x axes within each column, y axes within each row), `"columns"`, `"rows"`, `"all"` or `FALSE`. Tick labels are hidden on all but the outer axes.
`start_cell`	Which cell is row 1, column 1: `"top-left"` or `"bottom-left"`.
`print_grid`	If `TRUE`, print the grid layout (see `print_grid()`).
`horizontal_spacing`, `vertical_spacing`	Space between columns or rows, as a fraction of the figure width or height. Defaults to `0.2 / cols` (`0.4 / cols` with secondary y axes) and `0.3 / rows` (`0.5 / rows` with subplot titles).
`subplot_titles`	Character vector of titles, one per subplot in row-major order (insets last). Empty strings are skipped.
`column_widths`, `row_heights`	Relative column widths and row heights; default equal.
`specs`	Per-cell subplot specifications: a list of `rows` lists, each holding `cols` elements. Each element is `NULL` (empty cell) or a named list with any of: • `type`: `"xy"` (default, 2D cartesian), `"scene"`, `"polar"`, `"ternary"`, `"geo"`, `"map"`, `"domain"` (e.g. pie charts), or a trace type such as `"bar"` or `"pie"` to use its subplot type. • `secondary_y`: `TRUE` to add a second y axis on the right (`"xy"` only).

- `colspan`, `rowspan`: number of columns or rows the subplot spans.
- `l`, `r`, `b`, `t`: padding on each side, as a fraction of the figure.

`insets` List of inset subplots, each a named list with `cell` (`c(row, col)` of the cell it sits in), `type`, and `l`, `b` (offsets) and `w`, `h` (size, or `"to_end"`) as fractions of the cell.

`column_titles`, `row_titles`
Titles above each column and to the right of each row.

`x_title`, `y_title`
Titles below and to the left of the whole grid.

`figure` An existing `go_figure()` to add the subplots to.

`font` Font for the subplot, column, row and x/y titles: a named list or `go_layout_annotation_font()` object. Default `list(size = 16)`.

Value

A `go_figure` with the subplot grid in its layout.

See Also

`print_grid()`, `get_subplot()`

Examples

```
fig <- make_subplots(rows = 1, cols = 2, subplot_titles = c("Left", "Right")) |>
  add_scatter(x = 1:3, y = c(4, 5, 6), row = 1, col = 1) |>
  add_scatter(x = 20:22, y = c(50, 60, 70), row = 1, col = 2) |>
  update_yaxes(title_text = "y", row = 1, col = 1)

# Mixed subplot types and a secondary y axis
fig <- make_subplots(
  rows = 2, cols = 2,
  specs = list(
    list(list(secondary_y = TRUE), list(type = "domain")),
    list(list(colspan = 2), NULL)
  ),
  print_grid = TRUE
)
```

---

partial_bundle *Use a smaller plotly.js bundle*

Description

By default, figures load the full plotly.js bundle (about 4.8 MB). plotly.js also publishes partial bundles with fewer trace types, which make pages (and files saved with `htmlwidgets::saveWidget()`) much smaller:

Usage

```
partial_bundle(p, type = "auto", local = TRUE, minified = TRUE)
```

Arguments

p	A go_figure() or htmlwidget.
type	"auto", "full", or the name of a partial bundle.
local, minified	Unused: the bundles are always the minified ones shipped with the package (kept for compatibility with the plotly package).

Details

- "basic": bar, pie, scatter (about 1.2 MB)
- "cartesian": also box, contour, heatmap, histogram, histogram2d, histogram2dcontour, image, scatterternary, violin
- "finance": bar, candlestick, funnel, funnelarea, histogram, indicator, ohlc, pie, scatter, waterfall
- "geo": choropleth, scatter, scattergeo
- "gl2d": parcoords, scatter, scattergl, splom
- "gl3d": cone, isosurface, mesh3d, scatter, scatter3d, streamtube, surface, volume
- "map": choroplethmap, densitymap, scatter, scattermap

type = "auto" uses the smallest bundle with all the figure's trace types (including those in animation frames), or the full bundle if none has them all. Use options(plotly.core.bundle = "auto") to do this for every figure.

A page has a single plotly.js: when it shows several figures, they should use the same bundle (or the full one, which is used if any figure needs it).

partial_bundle() is an S3 generic, so packages can add methods for their own figure classes (the plotly package does for plot_ly() objects).

Value

The figure, which renders with that bundle.

Examples

```
fig <- go_figure(go_bar(x = c("a", "b"), y = 1:2)) |> partial_bundle()
```

plotly_json *JSON of a plot*

Description

Returns the JSON plotly.js receives for a plot.

Usage

```
plotly_json(p = last_plot(), jsonedit = interactive(), pretty = TRUE, ...)
```

Arguments

p	A go_figure() or anything as_widget() accepts (default: the last plot printed).
jsonedit	Show it in an interactive viewer (needs the listviewer package)?
pretty	Pretty-print the JSON?
...	Passed to listviewer::jsonedit().

Value

The JSON string (or a listviewer widget).

Examples

```
cat(plotly_json(go_figure(go_bar(y = 1:3)), jsonedit = FALSE))
```

plotly_obj-modify *Modify graph objects*

Description

Setting a property of a graph object (created by a go_*() constructor) validates the new value, just like the constructor does. Setting a property to NULL removes it.

Usage

```
## S3 replacement method for class 'plotly_obj'
x$name <- value

## S3 replacement method for class 'plotly_obj'
x[[i]] <- value
```

Arguments

x	A graph object.
name, i	Property name.
value	New value.

Value

The modified graph object.

Examples

```
m <- go_scatter_marker(size = 10)
m$color <- "red"
m[["symbol"]] <- "diamond"
try(m$size <- -1)
```

---

plotly_widget *Render a figure as an htmlwidget*

Description

`plotly_widget()` and `as_widget()` turn a figure into an htmlwidget, which is what printing a figure does. Use them to embed figures with htmltools or `htmlwidgets::saveWidget()`.

Usage

```
plotly_widget(
  fig,
  width = NULL,
  height = NULL,
  elementId = NULL,
  config = NULL,
  bundle = NULL
)

as_widget(x, ...)
```

Arguments

fig	A `go_figure()`, a figure from a `px_*()` function, or a list with `data` (list of traces), `layout`, `frames` and `config` elements.
width, height	Widget size in CSS units, or `NULL` for the layout's `width`/`height` or to fill the container.
elementId	Optional element id for the widget.
config	Plotly.js configuration options (e.g. `list(displayModeBar = FALSE)`), merged over those set with `config()`.

bundle	The plotly.js bundle to load: "full", "auto" (the smallest bundle with all the figure's trace types) or the name of a partial bundle (see partial_bundle()). By default, the one set with partial_bundle(), or the plotly.core.bundle option ("full" if unset).
x	A figure or htmlwidget. as_widget() is an S3 generic: the plotly package adds a method for its plot_ly() objects.
...	Passed to plotly_widget().

Details

If the figure's layout has no template, the default template (see default_template()) is applied.

Numeric arrays in traces are sent to the browser as base64-encoded typed arrays ({dtype, bdata} objects, as plotly.py does), which are smaller and faster to load than JSON numbers. JavaScript that reads x.data in htmlwidgets::onRender() sees these objects rather than arrays; read the plotted values from el._fullData instead. plotly_json() shows plain JSON. Vectors of whole numbers are sent as integers; see as_float32() to send other numbers in single precision. Dates and date-times are sent as numbers and turned back into "YYYY-MM-DD HH:MM:SS" strings (in the vector's time zone) in the browser before plotting.

Value

An object of class htmlwidget.

Examples

```
plotly_widget(list(data = list(list(type = "bar", x = 1:3, y = c(2, 5, 3)))))
```

---

plotlyjs_version *Version of the bundled plotly.js library*

Description

Version of the bundled plotly.js library

Usage

```
plotlyjs_version()
```

Value

A character string.

Examples

```
plotlyjs_version()
```

plotlyOutput *Shiny bindings for plotly figures*

Description

`plotlyOutput()` creates a plot output element in a Shiny UI; `renderPlotly()` renders a `go_figure()`, a figure from a `px_*()` function or a plotly htmlwidget (or a promise of one) into it.

Usage

```
plotlyOutput(
  outputId,
  width = "100%",
  height = "400px",
  inline = FALSE,
  reportTheme = TRUE,
  fill = !inline
)

renderPlotly(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId`	Output variable to read the plot from.
`width, height`	Size of the plot output, as a valid CSS unit (e.g. `"100%"`, `"400px"`).
`inline`	Use an inline (`span()`) container instead of a block one.
`reportTheme`	Report CSS colors and fonts to the server (see `shiny::getCurrentOutputInfo()`).
`fill`	Whether the plot fills its container in fillable layouts.
`expr`	An expression that returns a figure.
`env`	The environment in which to evaluate `expr`.
`quoted`	Is `expr` a quoted expression?

Value

`plotlyOutput()` returns a Shiny UI element; `renderPlotly()` a Shiny render function.

See Also

`event_data()`, `plotlyProxy()`

plotlyProxy *Modify a plotly figure in Shiny without re-rendering it*

Description

plotlyProxy() refers to a plot rendered with renderPlotly(); plotlyProxyInvoke() calls a plotly.js function on it, e.g. "restyle", "relayout", "update", "addTraces", "deleteTraces", "extendTraces" or "animate". Arguments are passed to the plotly.js function after the plot element (see https://plotly.com/javascript/plotlyjs-function-reference/); "reconfig" sets a new config.

Usage

```
plotlyProxy(
  outputId,
  session = shiny::getDefaultReactiveDomain(),
  deferUntilFlush = TRUE
)

plotlyProxyInvoke(p, method, ...)
```

Arguments

outputId	The output id of the plot.
session	The Shiny session.
deferUntilFlush	Send the message after Shiny's next flush (so the plot exists).
p	A proxy from plotlyProxy().
method	A plotly.js function name.
...	Arguments to the plotly.js function. Graph objects and data frames are serialised like figures (data frames as rows).

Value

The proxy, invisibly for plotlyProxyInvoke().

Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  ui <- shiny::fluidPage(shiny::actionButton("add", "Add a point"), plotlyOutput("plot"))
  server <- function(input, output, session) {
    output$plot <- renderPlotly(go_figure(go_scatter(y = c(1, 3, 2))))
    shiny::observeEvent(input$add, {
      plotlyProxy("plot", session) |>
       plotlyProxyInvoke("extendTraces", list(y = list(list(stats::runif(1)))), list(0))
    })
  }
```

```
  shiny::shinyApp(ui, server)
}
```

`print.go_figure` *Print or knit a figure*

Description

Printing a figure renders it with `plotly_widget()` in the RStudio viewer or a browser. In R Markdown and Quarto documents, figures are rendered inline: as interactive widgets in HTML output, sized by the chunk options `fig.width`/`fig.height` (`fig-width`/`fig-height` in Quarto) unless the layout sets a size; and, for other formats (PDF, Word, ...), as static images made with `save_image()` when a browser is available.

Usage

```
## S3 method for class 'go_figure'
print(x, ...)

## S3 method for class 'go_figure'
knit_print(x, ..., options = NULL)
```

Arguments

`x`	A `go_figure` object.
`...`	Passed on to the htmlwidget print or knit_print method.
`options`	Chunk options (from knitr).

Details

Set the chunk option `plotly.static = TRUE` to use static images in HTML output too, or `plotly.static = FALSE` to always use a widget.

Value

`x`, invisibly, for `print()`.

print.plotly_obj *Print a graph object*

Description

Prints the object's class and its properties as JSON.

Usage

```
## S3 method for class 'plotly_obj'
print(x, ...)
```

Arguments

x	A graph object.
...	Unused.

Value

x, invisibly.

print_grid *Print or query a figure's subplot grid*

Description

print_grid() prints the layout of a grid created by make_subplots(). get_subplot() returns the layout object(s) of one grid cell: a list with xaxis and yaxis for 2D cartesian subplots, the scene, polar, etc. object for other subplot types, or the x and y domain for "domain" subplots.

Usage

```
print_grid(fig)

get_subplot(fig, row, col, secondary_y = FALSE)
```

Arguments

fig	A figure created by make_subplots().
row, col	Grid cell (1-based).
secondary_y	If TRUE, the cell's secondary y axis.

Value

print_grid() returns fig invisibly; get_subplot() returns the subplot's layout objects, or NULL for an empty cell.

Examples

```
fig <- make_subplots(rows = 2, cols = 2, specs = list(
  list(list(), list(type = "polar")),
  list(list(colspan = 2), NULL)
))
print_grid(fig)
get_subplot(fig, 1, 2)
```

---

px_area *Stacked area chart*

Description

In a stacked area plot, each row of `data_frame` is represented as a vertex of a polyline mark in 2D space. The area between successive polylines is filled.

Usage

```
px_area(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  line_group = NULL,
  color = NULL,
  pattern_shape = NULL,
  symbol = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  text = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  pattern_shape_sequence = NULL,
  pattern_shape_map = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  markers = FALSE,
  orientation = NULL,
```

```
  groupnorm = NULL,
  log_x = FALSE,
  log_y = FALSE,
  range_x = NULL,
  range_y = NULL,
  line_shape = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
line_group	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to group rows of data_frame into lines.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
pattern_shape	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign pattern shapes to marks.
symbol	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.
hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data
: Column names (a character vector or a formula such as `~c(a, b)`), or a list of column names, formulas and vectors. Values from these columns are added to the traces' `customdata` (e.g. for use in Shiny events) without appearing in the hover tooltip.

text
: A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector appear in the figure as text labels.

facet_row
: A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.

facet_col
: A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.

facet_col_wrap
: Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if `facet_row` or a `marginal` is set.

facet_row_spacing
: Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

facet_col_spacing
: Spacing between facet columns, in paper units Default is 0.02.

animation_frame
: A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to provide object-constancy across animation frames: rows with matching `animation_groups` will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. `list(day = c("Thur", "Fri", "Sat", "Sun"))`. By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. `c(gdpPercap = "GDP per capita")`.

color_discrete_sequence
: Character vector of colors, e.g. `px_colors$qualitative$Set1`. Values of `color` are assigned colors in order (cycling), unless set in `color_discrete_map`. Defaults to the template's `colorway`.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

pattern_shape_sequence
: Strings should define valid plotly.js patterns-shapes. When pattern_shape is set, values in that column are assigned patterns-shapes by cycling through pattern_shape_sequence in the order described in category_orders, unless the value of pattern_shape is a key in pattern_shape_map.

pattern_shape_map
: Named character vector or list mapping values of pattern_shape to pattern shapes, overriding pattern_shape_sequence. px_identity() uses the values directly.

symbol_sequence
: Strings should define valid plotly.js symbols. When symbol is set, values in that column are assigned symbols by cycling through symbol_sequence in the order described in category_orders, unless the value of symbol is a key in symbol_map.

symbol_map
: Named character vector or list mapping values of symbol to marker symbols, overriding symbol_sequence. px_identity() uses the values of symbol directly.

markers
: If TRUE, markers are shown on lines.

orientation
: (default "v" if x and y are provided and both continuous or both categorical, otherwise "v"("h") if x(y) is categorical and y(x) is continuous, otherwise "v"("h") if only x(y) is provided)

groupnorm
: One of "fraction" or "percent". If "fraction", the value of each point is divided by the sum of all values at that location coordinate. "percent" is the same but multiplied by 100 to show percentages. NULL will stack up all values at each location coordinate.

log_x
: If TRUE, the x-axis is log-scaled in cartesian coordinates.

log_y
: If TRUE, the y-axis is log-scaled in cartesian coordinates.

range_x
: If provided, overrides auto-scaling on the x-axis in cartesian coordinates.

range_y
: If provided, overrides auto-scaling on the y-axis in cartesian coordinates.

line_shape
: One of "linear", "spline", "hv", "vh", "hvh", or "vhv"

title
: The figure title.

subtitle
: The figure subtitle.

template
: A template name (see template_names()) or a template list. Defaults to default_template().

width
: The figure width in pixels.

height
: The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_gapminder() |>
  px_area(x = ~year, y = ~pop, color = ~continent, line_group = ~country)
```

---

px_bar *Bar chart*

Description

In a bar plot, each row of `data_frame` is represented as a rectangular mark.

Usage

```
px_bar(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  color = NULL,
  pattern_shape = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  text = NULL,
  base = NULL,
  error_x = NULL,
  error_x_minus = NULL,
  error_y = NULL,
  error_y_minus = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
```

```
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  color_continuous_scale = NULL,
  pattern_shape_sequence = NULL,
  pattern_shape_map = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  opacity = NULL,
  orientation = NULL,
  barmode = "relative",
  log_x = FALSE,
  log_y = FALSE,
  range_x = NULL,
  range_y = NULL,
  text_auto = FALSE,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
pattern_shape	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign pattern shapes to marks.
facet_row	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.
facet_col	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values

from this column or vector are used to assign marks to facetted subplots in the horizontal direction.

`facet_col_wrap` Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if `facet_row` or a `marginal` is set.

`facet_row_spacing`
Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

`facet_col_spacing`
Spacing between facet columns, in paper units Default is 0.02.

`hover_name` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector appear in bold in the hover tooltip.

`hover_data` Extra data to show in the hover tooltip: column names (a character vector or a formula such as `~c(a, b)`), or a named list with column names as names and values `TRUE` (default formatting), `FALSE` (hide the column, e.g. one shown by default), a d3-format string such as `":.3f"`, a vector of data, or a list of a format string or `TRUE` and a vector of data.

`custom_data` Column names (a character vector or a formula such as `~c(a, b)`), or a list of column names, formulas and vectors. Values from these columns are added to the traces' `customdata` (e.g. for use in Shiny events) without appearing in the hover tooltip.

`text` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector appear in the figure as text labels.

`base` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to position the base of the bar.

`error_x` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to size x-axis error bars. If `error_x_minus` is `NULL`, error bars will be symmetrical, otherwise `error_x` is used for the positive direction only.

`error_x_minus` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to size x-axis error bars in the negative direction. Ignored if `error_x` is `NULL`.

`error_y` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to size y-axis error bars. If `error_y_minus` is `NULL`, error bars will be symmetrical, otherwise `error_y` is used for the positive direction only.

`error_y_minus` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to size y-axis error bars in the negative direction. Ignored if `error_y` is `NULL`.

animation_frame
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

color_continuous_scale
: Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

pattern_shape_sequence
: Strings should define valid plotly.js patterns-shapes. When pattern_shape is set, values in that column are assigned patterns-shapes by cycling through pattern_shape_sequence in the order described in category_orders, unless the value of pattern_shape is a key in pattern_shape_map.

pattern_shape_map
: Named character vector or list mapping values of pattern_shape to pattern shapes, overriding pattern_shape_sequence. px_identity() uses the values directly.

range_color
: If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
: If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

opacity
: Value between 0 and 1. Sets the opacity for markers.

orientation
: (default "v" if x and y are provided and both continuous or both categorical, otherwise "v"("h") if x(y) is categorical and y(x) is continuous, otherwise "v"("h") if only x(y) is provided)

barmode	One of "group", "overlay" or "relative" In "relative" mode, bars are stacked above zero for positive values and below zero for negative values. In "overlay" mode, bars are drawn on top of one another. In "group" mode, bars are placed beside each other.
log_x	If TRUE, the x-axis is log-scaled in cartesian coordinates.
log_y	If TRUE, the y-axis is log-scaled in cartesian coordinates.
range_x	If provided, overrides auto-scaling on the x-axis in cartesian coordinates.
range_y	If provided, overrides auto-scaling on the y-axis in cartesian coordinates.
text_auto	If TRUE or a string, the x or y or z values will be displayed as text, depending on the orientation A string like ".2f" will be interpreted as a texttemplate numeric formatting directive.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_tips() |> px_bar(x = ~day, y = ~total_bill, color = ~sex, barmode = "group")
```

---

px_bar_polar *Polar bar chart*

Description

In a polar bar plot, each row of data_frame is represented as a wedge mark in polar coordinates.

Usage

```
px_bar_polar(
  data_frame = NULL,
  r = NULL,
  theta = NULL,
  color = NULL,
  pattern_shape = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  base = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  color_continuous_scale = NULL,
  pattern_shape_sequence = NULL,
  pattern_shape_map = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  barnorm = NULL,
  barmode = "relative",
  direction = "clockwise",
  start_angle = 90,
  range_r = NULL,
  range_theta = NULL,
  log_r = FALSE,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
r	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the radial axis in polar coordinates.
theta	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values

from this column or vector are used to position marks along the angular axis in polar coordinates.

color A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.

pattern_shape A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign pattern shapes to marks.

hover_name A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

base A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position the base of the bar.

animation_frame
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

color_continuous_scale
: Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

pattern_shape_sequence
: Strings should define valid plotly.js patterns-shapes. When pattern_shape is set, values in that column are assigned patterns-shapes by cycling through pattern_shape_sequence in the order described in category_orders, unless the value of pattern_shape is a key in pattern_shape_map.

pattern_shape_map
: Named character vector or list mapping values of pattern_shape to pattern shapes, overriding pattern_shape_sequence. px_identity() uses the values directly.

range_color
: If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
: If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

barnorm
: One of "fraction" or "percent". If "fraction", the value of each bar is divided by the sum of all values at that location coordinate. "percent" is the same but multiplied by 100 to show percentages. NULL will stack up all values at each location coordinate.

barmode
: One of "group", "overlay" or "relative" In "relative" mode, bars are stacked above zero for positive values and below zero for negative values. In "overlay" mode, bars are drawn on top of one another. In "group" mode, bars are placed beside each other.

direction
: One of "counterclockwise" or "clockwise". Default is "clockwise" Sets the direction in which increasing values of the angular axis are drawn.

start_angle
: Sets start angle for the angular axis, with 0 being due east and 90 being due north.

range_r
: If provided, overrides auto-scaling on the radial axis in polar coordinates.

range_theta
: If provided, overrides auto-scaling on the angular axis in polar coordinates.

log_r
: If TRUE, the radial axis is log-scaled in polar coordinates.

title
: The figure title.

subtitle
: The figure subtitle.

template
: A template name (see template_names()) or a template list. Defaults to default_template().

width
: The figure width in pixels.

height
: The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_wind() |> px_bar_polar(r = ~frequency, theta = ~direction, color = ~strength)
```

---

px_box *Box plot*

Description

In a box plot, rows of data_frame are grouped together into a box-and-whisker mark to visualize their distribution. Each box spans from quartile 1 (Q1) to quartile 3 (Q3). The second quartile (Q2) is marked by a line inside the box. By default, the whiskers correspond to the box' edges +/- 1.5 times the interquartile range (IQR: Q3-Q1), see "points" for other options.

Usage

```
px_box(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  color = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
```

```
  color_discrete_map = NULL,
  orientation = NULL,
  boxmode = NULL,
  log_x = FALSE,
  log_y = FALSE,
  range_x = NULL,
  range_y = NULL,
  points = NULL,
  notched = FALSE,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
facet_row	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.
facet_col	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.
facet_col_wrap	Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.
facet_row_spacing	Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

facet_col_spacing
: Spacing between facet columns, in paper units Default is 0.02.

hover_name
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data
: Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data
: Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

animation_frame
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

orientation
: (default "v" if x and y are provided and both continuous or both categorical, otherwise "v"("h") if x(y) is categorical and y(x) is continuous, otherwise "v"("h") if only x(y) is provided)

boxmode
: One of "group" or "overlay" In "overlay" mode, boxes are on drawn top of one another. In "group" mode, boxes are placed beside each other.

log_x
: If TRUE, the x-axis is log-scaled in cartesian coordinates.

log_y	If TRUE, the y-axis is log-scaled in cartesian coordinates.
range_x	If provided, overrides auto-scaling on the x-axis in cartesian coordinates.
range_y	If provided, overrides auto-scaling on the y-axis in cartesian coordinates.
points	One of "outliers", "suspectedoutliers", "all", or FALSE. If "outliers", only the sample points lying outside the whiskers are shown. If "suspectedoutliers", all outlier points are shown and those less than 4*Q1-3*Q3 or greater than 4*Q3-3*Q1 are highlighted with the marker's "outliercolor". If "outliers", only the sample points lying outside the whiskers are shown. If "all", all sample points are shown. If FALSE, no sample points are shown and the whiskers extend to the full range of the sample.
notched	If TRUE, boxes are drawn with notches.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_tips() |> px_box(x = ~time, y = ~total_bill, points = "all")
```

---

px_choropleth *Choropleth map*

Description

In a choropleth map, each row of data_frame is represented by a colored region mark on a map.

Usage

```
px_choropleth(
  data_frame = NULL,
  lat = NULL,
  lon = NULL,
  locations = NULL,
  locationmode = NULL,
  geojson = NULL,
  featureidkey = NULL,
  color = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  projection = NULL,
  scope = NULL,
  center = NULL,
  fitbounds = NULL,
  basemap_visible = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
lat	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks according to latitude on a map.

lon — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks according to longitude on a map.

locations — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are to be interpreted according to locationmode and mapped to longitude/latitude.

locationmode — One of "ISO-3", "USA-states", or 'country names' Determines the set of locations used to match entries in locations to regions on the map.

geojson — GeoJSON as a list (e.g. from jsonlite::read_json()), with features whose ids are referenced by locations.

featureidkey — Path to field in GeoJSON feature object with which to match the values passed in to locations.The most common alternative to the default is of the form 'properties.<key>.

color — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.

facet_row — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.

facet_col — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.

facet_col_wrap — Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.

facet_row_spacing — Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

facet_col_spacing — Spacing between facet columns, in paper units Default is 0.02.

hover_name — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data — Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data — Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

animation_frame
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

color_continuous_scale
: Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color
: If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
: If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

projection
: One of "equirectangular", "mercator", "orthographic", "natural earth", "kavrayskiy7", "miller", "robinson", "eckert4", "azimuthal equal area", "azimuthal equidistant", "conic equal area", "conic conformal", "conic equidistant", "gnomonic", "stereographic", "mollweide", "hammer", "transverse mercator", "albers usa", "winkel tripel", "aitoff", or "sinusoidal"Default depends on scope.

scope
: One of "world", "usa", "europe", "asia", "africa", "north america", or "south america"Default is "world" unless projection is set to "albers usa", which forces "usa".

center
: Dict keys are "lat" and "lon" Sets the center point of the map.

fitbounds
: One of FALSE, locations or geojson.

basemap_visible
: Force the basemap visibility.

title
: The figure title.

subtitle
: The figure subtitle.

template
: A template name (see template_names()) or a template list. Defaults to default_template().

width
: The figure width in pixels.

height
: The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_gapminder(year = 2007) |>
  px_choropleth(locations = ~iso_alpha, color = ~lifeExp, hover_name = ~country)
```

---

px_choropleth_map *Choropleth map (tile-based)*

Description

In a choropleth map, each row of data_frame is represented by a colored region on the map.

Usage

```
px_choropleth_map(
  data_frame = NULL,
  geojson = NULL,
  featureidkey = NULL,
  locations = NULL,
  color = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  animation_frame = NULL,
```

```
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  opacity = NULL,
  zoom = NULL,
  center = NULL,
  map_style = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
geojson	GeoJSON as a list (e.g. from jsonlite::read_json()), with features whose ids are referenced by locations.
featureidkey	Path to field in GeoJSON feature object with which to match the values passed in to locations.The most common alternative to the default is of the form 'properties.<key>.
locations	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are to be interpreted according to locationmode and mapped to longitude/latitude.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.
hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.
custom_data	Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to

the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

animation_frame
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

color_continuous_scale
: Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color
: If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
: If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

opacity
: Value between 0 and 1. Sets the opacity for markers.

zoom
: Between 0 and 20. Sets initial map zoom level. Higher values are more zoomed-in. If not provided, the map will select its own zoom level.

center
: Dict keys are "lat" and "lon" Sets the center point of the map.

map_style
: Identifier of base map style. Allowed values are "basic", "carto-darkmatter", "carto-darkmatter-nolabels", "carto-positron", "carto-positron-nolabels", "carto-voyager", "carto-voyager-nolabels", "dark", "light", "open-street-map", "outdoors", "satellite", "satellite-streets", "streets", "white-bg".

title
: The figure title.

subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_election() |>
  px_choropleth_map(geojson = px_data_election_geojson(), locations = ~district,
                    featureidkey = "properties.district", color = ~Bergeron,
                    center = list(lat = 45.5517, lon = -73.7073), zoom = 9)
```

---

px_colors *Color sequences and scales*

Description

The named color sequences and scales of plotly.py's plotly.colors module: px_colors$qualitative (discrete color sequences, e.g. Plotly, D3, Set1), px_colors$sequential (e.g. Viridis, Plasma, Blues), px_colors$diverging (e.g. RdBu) and px_colors$cyclical (e.g. Twilight). Each is a character vector of colors; use rev() to reverse one.

Usage

```
px_colors
```

Format

A list of four named lists of character vectors.

Details

Sequential, diverging and cyclical scales can also be given by name (case-insensitive, with "_r" appended to reverse) to any colorscale property, e.g. marker = list(colorscale = "viridis_r").

Examples

```
names(px_colors$qualitative)
px_colors$qualitative$Plotly
rev(px_colors$sequential$Viridis)
```

`px_constant` *Constant and range inputs for Plotly Express*

Description

Pass `px_constant()` to an argument that expects a column, to use the same value for every row (with an optional legend/axis `label`). `px_range()` uses the integers 0, 1, 2, ... `px_identity()` can be passed to `color_discrete_map`, `symbol_map`, `line_dash_map` or `pattern_shape_map` to use the data values themselves (e.g. colors).

Usage

```
px_constant(value, label = NULL)

px_range(label = NULL)

px_identity()
```

Arguments

`value`	The constant value.
`label`	Name used for the column in labels and hover text.

Value

An object recognised by the `px_*()` functions.

Examples

```
px_data_iris() |>
  px_scatter(x = ~sepal_width, y = ~sepal_length, color = px_constant("all flowers"))
px_data_iris() |> px_scatter(x = px_range("index"), y = ~sepal_length)
```

px_data *Built-in datasets*

Description

Datasets used in the plotly.express examples, returned as data.tables.

Usage

```
px_data_gapminder(
  datetimes = FALSE,
  centroids = FALSE,
  year = NULL,
  pretty_names = FALSE
)

px_data_tips(pretty_names = FALSE)

px_data_iris()

px_data_wind()

px_data_election()

px_data_election_geojson()

px_data_carshare()

px_data_stocks(datetimes = FALSE)

px_data_experiment()

px_data_medals_wide()

px_data_medals_long()
```

Arguments

datetimes	Convert the date column to Date.
centroids	Keep country centroid columns.
year	Keep only this year.
pretty_names	Use readable column names.

Details

- px_data_gapminder(): life expectancy, population and GDP per capita by country and year (1704 rows). datetimes = TRUE makes year a Date; centroids = TRUE keeps the centroid_lat

and centroid_lon columns; year filters to one year; pretty_names = TRUE uses readable column names.

- px_data_tips(): restaurant bills and tips (244 rows).
- px_data_iris(): Fisher's iris measurements, with a numeric species_id (150 rows).
- px_data_wind(): wind direction and strength frequencies (128 rows).
- px_data_election(): results of the 2013 Montreal mayoral election by district (58 rows); px_data_election_geojson() returns the district boundaries as a GeoJSON list.
- px_data_carshare(): car-sharing availability in Montreal (249 rows).
- px_data_stocks(): weekly closing prices of six technology stocks, normalised to 1 on 2018-01-01 (105 rows). datetimes = TRUE makes date a Date.
- px_data_experiment(): results of a fictional experiment (100 rows).
- px_data_medals_wide(), px_data_medals_long(): medal counts by nation in wide and long form.

Value

A data.table (a GeoJSON list for px_data_election_geojson()).

Examples

```
head(px_data_gapminder(year = 2007))
head(px_data_tips())
```

---

px_defaults *Plotly Express defaults*

Description

px_defaults is an environment of default values used by the px_*() functions when the corresponding argument is NULL, like plotly.py's px.defaults: template, width, height, color_discrete_sequence, color_discrete_map, color_continuous_scale, symbol_sequence, symbol_map, line_dash_sequence, line_dash_map, pattern_shape_sequence, pattern_shape_map, size_max, category_orders and labels. reset_px_defaults() restores the initial values.

Usage

```
px_defaults

reset_px_defaults()
```

Format

An environment.

Value

`reset_px_defaults()` returns `px_defaults` invisibly.

Examples

```
px_defaults$template <- "plotly_white"
px_defaults$color_continuous_scale <- px_colors$sequential$Blackbody
reset_px_defaults()
```

---

px_density_contour *Density contour plot*

Description

In a density contour plot, rows of `data_frame` are grouped together into contour marks to visualize the 2D distribution of an aggregate function `histfunc` (e.g. the count or sum) of the value `z`.

Usage

```
px_density_contour(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  z = NULL,
  color = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  orientation = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  marginal_x = NULL,
  marginal_y = NULL,
  trendline = NULL,
  trendline_options = NULL,
  trendline_color_override = NULL,
  trendline_scope = "trace",
  log_x = FALSE,
  log_y = FALSE,
```

```
  range_x = NULL,
  range_y = NULL,
  histfunc = NULL,
  histnorm = NULL,
  nbinsx = NULL,
  nbinsy = NULL,
  text_auto = FALSE,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
z	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the z axis in cartesian coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
facet_row	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.
facet_col	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.
facet_col_wrap	Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.

facet_row_spacing
: Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

facet_col_spacing
: Spacing between facet columns, in paper units Default is 0.02.

hover_name
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data
: Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

animation_frame
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

orientation
: (default "v" if x and y are provided and both continuous or both categorical, otherwise "v"("h") if x(y) is categorical and y(x) is continuous, otherwise "v"("h") if only x(y) is provided)

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

marginal_x
: One of "rug", "box", "violin", or "histogram". If set, a horizontal subplot is drawn above the main plot, visualizing the x-distribution.

marginal_y
: One of "rug", "box", "violin", or "histogram". If set, a vertical subplot is drawn to the right of the main plot, visualizing the y-distribution.

trendline — One of "ols" (ordinary least squares regression), "lowess" (locally weighted smoothing), "rolling", "expanding" or "ewm" (exponentially weighted moving average). A trendline is drawn for each discrete color/symbol group; configure it with trendline_options. Fitted OLS models are available from px_get_trendline_results().

trendline_options — Named list of options for the trendline function: for "ols", add_constant, log_x and log_y; for "lowess", frac; for "rolling", window, min_periods, center and function ("mean", "median", "sum", "min", "max", "std", "var", "count"); for "expanding", min_periods and function; for "ewm", one of com, span, halflife or alpha, and adjust.

trendline_color_override — Valid CSS color. If provided, and if trendline is set, all trendlines will be drawn in this color rather than in the same color as the traces from which they draw their inputs.

trendline_scope — If "trace", then one trendline is drawn per trace (i.e. per color, symbol, facet, animation frame etc) and if "overall" then one trendline is computed for the entire dataset, and replicated across all facets.

log_x — If TRUE, the x-axis is log-scaled in cartesian coordinates.

log_y — If TRUE, the y-axis is log-scaled in cartesian coordinates.

range_x — If provided, overrides auto-scaling on the x-axis in cartesian coordinates.

range_y — If provided, overrides auto-scaling on the y-axis in cartesian coordinates.

histfunc — One of "count", "sum", "avg", "min", or "max". Function used to aggregate values for summarization (note: can be normalized with histnorm).

histnorm — One of "percent", "probability", "density", or "probability density" If NULL, the output of histfunc is used as is. If "probability", the output of histfunc for a given bin is divided by the sum of the output of histfunc for all bins. If "percent", the output of histfunc for a given bin is divided by the sum of the output of histfunc for all bins and multiplied by 100. If "density", the output of histfunc for a given bin is divided by the size of the bin. If "probability density", the output of histfunc for a given bin is normalized such that it corresponds to the probability that a random event whose distribution is described by the output of histfunc will fall into that bin.

nbinsx — Positive integer. Sets the number of bins along the x axis.

nbinsy — Positive integer. Sets the number of bins along the y axis.

text_auto — If TRUE or a string, the x or y or z values will be displayed as text, depending on the orientation A string like ".2f" will be interpreted as a texttemplate numeric formatting directive.

title — The figure title.

subtitle — The figure subtitle.

template — A template name (see template_names()) or a template list. Defaults to default_template().

width — The figure width in pixels.

height — The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_tips() |> px_density_contour(x = ~total_bill, y = ~tip)
```

---

px_density_heatmap *Density heatmap*

Description

In a density heatmap, rows of data_frame are grouped together into colored rectangular tiles to visualize the 2D distribution of an aggregate function histfunc (e.g. the count or sum) of the value z.

Usage

```
px_density_heatmap(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  z = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  orientation = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
```

```
  color_continuous_midpoint = NULL,
  marginal_x = NULL,
  marginal_y = NULL,
  opacity = NULL,
  log_x = FALSE,
  log_y = FALSE,
  range_x = NULL,
  range_y = NULL,
  histfunc = NULL,
  histnorm = NULL,
  nbinsx = NULL,
  nbinsy = NULL,
  text_auto = FALSE,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
z	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the z axis in cartesian coordinates.
facet_row	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.
facet_col	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.
facet_col_wrap	Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.

facet_row_spacing
: Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

facet_col_spacing
: Spacing between facet columns, in paper units Default is 0.02.

hover_name
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data
: Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

animation_frame
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

orientation
: (default "v" if x and y are provided and both continuous or both categorical, otherwise "v"("h") if x(y) is categorical and y(x) is continuous, otherwise "v"("h") if only x(y) is provided)

color_continuous_scale
: Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color
: If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
: If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

marginal_x
: One of "rug", "box", "violin", or "histogram". If set, a horizontal subplot is drawn above the main plot, visualizing the x-distribution.

marginal_y
: One of "rug", "box", "violin", or "histogram". If set, a vertical subplot is drawn to the right of the main plot, visualizing the y-distribution.

opacity	Value between 0 and 1. Sets the opacity for markers.
log_x	If TRUE, the x-axis is log-scaled in cartesian coordinates.
log_y	If TRUE, the y-axis is log-scaled in cartesian coordinates.
range_x	If provided, overrides auto-scaling on the x-axis in cartesian coordinates.
range_y	If provided, overrides auto-scaling on the y-axis in cartesian coordinates.
histfunc	One of "count", "sum", "avg", "min", or "max". Function used to aggregate values for summarization (note: can be normalized with histnorm).
histnorm	One of "percent", "probability", "density", or "probability density" If NULL, the output of histfunc is used as is. If "probability", the output of histfunc for a given bin is divided by the sum of the output of histfunc for all bins. If "percent", the output of histfunc for a given bin is divided by the sum of the output of histfunc for all bins and multiplied by 100. If "density", the output of histfunc for a given bin is divided by the size of the bin. If "probability density", the output of histfunc for a given bin is normalized such that it corresponds to the probability that a random event whose distribution is described by the output of histfunc will fall into that bin.
nbinsx	Positive integer. Sets the number of bins along the x axis.
nbinsy	Positive integer. Sets the number of bins along the y axis.
text_auto	If TRUE or a string, the x or y or z values will be displayed as text, depending on the orientation A string like ".2f" will be interpreted as a texttemplate numeric formatting directive.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_tips() |> px_density_heatmap(x = ~total_bill, y = ~tip, marginal_x = "histogram")
```

px_density_map *Density map*

Description

In a density map, each row of `data_frame` contributes to the intensity of the color of the region around the corresponding point on the map.

Usage

```
px_density_map(
  data_frame = NULL,
  lat = NULL,
  lon = NULL,
  z = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  opacity = NULL,
  zoom = NULL,
  center = NULL,
  map_style = NULL,
  radius = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

`data_frame`	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if `NULL`, a data frame is built from the vectors given in the other arguments.
`lat`	A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to position marks according to latitude on a map.

lon — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks according to longitude on a map.

z — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the z axis in cartesian coordinates.

hover_name — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data — Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data — Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

animation_frame — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders — Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels — Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_continuous_scale — Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color — If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint — If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

opacity	Value between 0 and 1. Sets the opacity for markers.
zoom	Between 0 and 20. Sets initial map zoom level. Higher values are more zoomed-in. If not provided, the map will select its own zoom level.
center	Dict keys are "lat" and "lon" Sets the center point of the map.
map_style	Identifier of base map style. Allowed values are "basic", "carto-darkmatter", "carto-darkmatter-nolabels", "carto-positron", "carto-positron-nolabels", "carto-voyager", "carto-voyager-nolabels", "dark", "light", "open-street-map", "outdoors", "satellite", "satellite-streets", "streets", "white-bg".
radius	Sets the radius of influence of each point.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_carshare() |>
  px_density_map(lat = ~centroid_lat, lon = ~centroid_lon, z = ~car_hours, radius = 10)
```

---

px_ecdf *Empirical cumulative distribution*

Description

In a Empirical Cumulative Distribution Function (ECDF) plot, rows of data_frame are sorted by the value x (or y if orientation is "h") and their cumulative count (or the cumulative sum of y if supplied and orientation is h) is drawn as a line.

Usage

```
px_ecdf(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  color = NULL,
  text = NULL,
  line_dash = NULL,
  symbol = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  markers = FALSE,
  lines = TRUE,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  line_dash_sequence = NULL,
  line_dash_map = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  marginal = NULL,
  opacity = NULL,
  orientation = NULL,
  ecdfnorm = "probability",
  ecdfmode = "standard",
  render_mode = "auto",
  log_x = FALSE,
  log_y = FALSE,
  range_x = NULL,
  range_y = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
text	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.
line_dash	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign dash-patterns to lines.
symbol	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.
facet_row	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.
facet_col	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.
facet_col_wrap	Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.
facet_row_spacing	Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.
facet_col_spacing	Spacing between facet columns, in paper units Default is 0.02.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

animation_frame
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

markers If TRUE, markers are shown on lines.

lines If FALSE, lines are not drawn (forced to TRUE if markers is FALSE).

category_orders
Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

line_dash_sequence
Strings should define valid plotly.js dash-patterns. When line_dash is set, values in that column are assigned dash-patterns by cycling through line_dash_sequence in the order described in category_orders, unless the value of line_dash is a key in line_dash_map.

line_dash_map Named character vector or list mapping values of line_dash to dash patterns, overriding line_dash_sequence. px_identity() uses the values of line_dash directly.

symbol_sequence
Strings should define valid plotly.js symbols. When symbol is set, values in that column are assigned symbols by cycling through symbol_sequence in the order described in category_orders, unless the value of symbol is a key in symbol_map.

symbol_map	Named character vector or list mapping values of symbol to marker symbols, overriding symbol_sequence. px_identity() uses the values of symbol directly.
marginal	One of "rug", "box", "violin", or "histogram". If set, a subplot is drawn alongside the main plot, visualizing the distribution.
opacity	Value between 0 and 1. Sets the opacity for markers.
orientation	(default "v" if x and y are provided and both continuous or both categorical, otherwise "v"("h") if x(y) is categorical and y(x) is continuous, otherwise "v"("h") if only x(y) is provided)
ecdfnorm	One of "probability" or "percent" If NULL, values will be raw counts or sums. If "probability", values will be probabilities normalized from 0 to 1. If "percent", values will be percentages normalized from 0 to 100.
ecdfmode	One of "standard", "complementary" or "reversed" If "standard", the ECDF is plotted such that values represent data at or below the point. If "complementary", the CCDF is plotted such that values represent data above the point. If "reversed", a variant of the CCDF is plotted such that values represent data at or above the point.
render_mode	One of "auto", "svg" or "webgl", default "auto" Controls the browser API used to draw marks. "svg" is appropriate for figures of less than 1000 data points, and will allow for fully-vectorized output. "webgl" is likely necessary for acceptable performance above 1000 points but rasterizes part of the output. "auto" uses heuristics to choose the mode.
log_x	If TRUE, the x-axis is log-scaled in cartesian coordinates.
log_y	If TRUE, the y-axis is log-scaled in cartesian coordinates.
range_x	If provided, overrides auto-scaling on the x-axis in cartesian coordinates.
range_y	If provided, overrides auto-scaling on the y-axis in cartesian coordinates.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_tips() |> px_ecdf(x = ~total_bill, color = ~sex)
```

`px_funnel` *Funnel chart*

Description

In a funnel plot, each row of `data_frame` is represented as a rectangular sector of a funnel.

Usage

```
px_funnel(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  color = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  text = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  opacity = NULL,
  orientation = NULL,
  log_x = FALSE,
  log_y = FALSE,
  range_x = NULL,
  range_y = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
facet_row	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.
facet_col	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.
facet_col_wrap	Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.
facet_row_spacing	Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.
facet_col_spacing	Spacing between facet columns, in paper units Default is 0.02.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.
hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.
custom_data	Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.
text	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.

animation_frame
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

opacity
: Value between 0 and 1. Sets the opacity for markers.

orientation
: (default "v" if x and y are provided and both continuous or both categorical, otherwise "v"("h") if x(y) is categorical and y(x) is continuous, otherwise "v"("h") if only x(y) is provided)

log_x
: If TRUE, the x-axis is log-scaled in cartesian coordinates.

log_y
: If TRUE, the y-axis is log-scaled in cartesian coordinates.

range_x
: If provided, overrides auto-scaling on the x-axis in cartesian coordinates.

range_y
: If provided, overrides auto-scaling on the y-axis in cartesian coordinates.

title
: The figure title.

subtitle
: The figure subtitle.

template
: A template name (see template_names()) or a template list. Defaults to default_template().

width
: The figure width in pixels.

height
: The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
data.frame(stage = c("Visit", "Download", "Sign up", "Purchase"),
           number = c(39, 27.4, 20.6, 11)) |>
  px_funnel(x = ~number, y = ~stage)
```

---

px_funnel_area *Funnel area chart*

Description

In a funnel area plot, each row of `data_frame` is represented as a trapezoidal sector of a funnel.

Usage

```
px_funnel_area(
  data_frame = NULL,
  names = NULL,
  values = NULL,
  color = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  labels = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL,
  opacity = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
names	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used as labels for sectors.
values	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to set values associated to sectors.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
color_discrete_sequence	Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.
color_discrete_map	Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.
hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.
custom_data	Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.
labels	Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.
opacity	Value between 0 and 1. Sets the opacity for markers.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
data.frame(stage = c("Visit", "Download", "Sign up", "Purchase"),
           number = c(39, 27.4, 20.6, 11)) |>
  px_funnel_area(values = ~number, names = ~stage)
```

---

px_get_trendline_results

Trendline fit results

Description

Returns the fitted models behind the OLS trendlines of a figure created by a px_*() function with trendline = "ols", one row per trendline, like plotly.py's px.get_trendline_results().

Usage

```
px_get_trendline_results(fig)
```

Arguments

fig	A figure returned by px_scatter() or a similar function.

Value

A data.table with a column per grouping variable and a list column px_fit_results of stats::lm() fits (NULL for other trendline types).

Examples

```
fig <- px_data_tips() |>
  px_scatter(x = ~total_bill, y = ~tip, color = ~smoker, trendline = "ols")
res <- px_get_trendline_results(fig)
summary(res$px_fit_results[[1]])
```

px_histogram *Histogram*

Description

In a histogram, rows of `data_frame` are grouped together into a rectangular mark to visualize the 1D distribution of an aggregate function `histfunc` (e.g. the count or sum) of the value `y` (or `x` if `orientation` is `"h"`).

Usage

```
px_histogram(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  color = NULL,
  pattern_shape = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  pattern_shape_sequence = NULL,
  pattern_shape_map = NULL,
  marginal = NULL,
  opacity = NULL,
  orientation = NULL,
  barmode = "relative",
  barnorm = NULL,
  histnorm = NULL,
  log_x = FALSE,
  log_y = FALSE,
  range_x = NULL,
  range_y = NULL,
  histfunc = NULL,
  cumulative = NULL,
  nbins = NULL,
  text_auto = FALSE,
  title = NULL,
```

```
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
pattern_shape	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign pattern shapes to marks.
facet_row	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.
facet_col	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.
facet_col_wrap	Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.
facet_row_spacing	Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.
facet_col_spacing	Spacing between facet columns, in paper units Default is 0.02.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.
hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and

values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

animation_frame
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

pattern_shape_sequence
: Strings should define valid plotly.js patterns-shapes. When pattern_shape is set, values in that column are assigned patterns-shapes by cycling through pattern_shape_sequence in the order described in category_orders, unless the value of pattern_shape is a key in pattern_shape_map.

pattern_shape_map
: Named character vector or list mapping values of pattern_shape to pattern shapes, overriding pattern_shape_sequence. px_identity() uses the values directly.

marginal
: One of "rug", "box", "violin", or "histogram". If set, a subplot is drawn alongside the main plot, visualizing the distribution.

opacity
: Value between 0 and 1. Sets the opacity for markers.

orientation
: (default "v" if x and y are provided and both continuous or both categorical, otherwise "v"("h") if x(y) is categorical and y(x) is continuous, otherwise "v"("h") if only x(y) is provided)

barmode
: One of "group", "overlay" or "relative" In "relative" mode, bars are stacked above zero for positive values and below zero for negative values. In "overlay" mode, bars are drawn on top of one another. In "group" mode, bars are placed beside each other.

barnorm	One of "fraction" or "percent". If "fraction", the value of each bar is divided by the sum of all values at that location coordinate. "percent" is the same but multiplied by 100 to show percentages. NULL will stack up all values at each location coordinate.
histnorm	One of "percent", "probability", "density", or "probability density" If NULL, the output of histfunc is used as is. If "probability", the output of histfunc for a given bin is divided by the sum of the output of histfunc for all bins. If "percent", the output of histfunc for a given bin is divided by the sum of the output of histfunc for all bins and multiplied by 100. If "density", the output of histfunc for a given bin is divided by the size of the bin. If "probability density", the output of histfunc for a given bin is normalized such that it corresponds to the probability that a random event whose distribution is described by the output of histfunc will fall into that bin.
log_x	If TRUE, the x-axis is log-scaled in cartesian coordinates.
log_y	If TRUE, the y-axis is log-scaled in cartesian coordinates.
range_x	If provided, overrides auto-scaling on the x-axis in cartesian coordinates.
range_y	If provided, overrides auto-scaling on the y-axis in cartesian coordinates.
histfunc	One of "count", "sum", "avg", "min", or "max". Function used to aggregate values for summarization (note: can be normalized with histnorm).
cumulative	If TRUE, histogram values are cumulative.
nbins	Positive integer. Sets the number of bins.
text_auto	If TRUE or a string, the x or y or z values will be displayed as text, depending on the orientation A string like ".2f" will be interpreted as a texttemplate numeric formatting directive.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_tips() |> px_histogram(x = ~total_bill, color = ~sex, marginal = "rug", nbins = 20)
```

`px_icicle` *Icicle chart*

Description

An icicle plot represents hierarchial data with adjoined rectangular sectors that all cascade from root down to leaf in one direction.

Usage

```
px_icicle(
  data_frame = NULL,
  names = NULL,
  values = NULL,
  parents = NULL,
  path = NULL,
  ids = NULL,
  color = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  labels = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL,
  branchvalues = NULL,
  maxdepth = NULL
)
```

Arguments

`data_frame`	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if `NULL`, a data frame is built from the vectors given in the other arguments.
`names`	A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used as labels for sectors.

values A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to set values associated to sectors.

parents A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used as parents in sunburst and treemap charts.

path Column names in data_frame (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. The columns define the hierarchy of sectors, from root to leaves. Cannot be combined with ids or parents.

ids A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to set ids of sectors

color A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.

color_continuous_scale
Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

color_discrete_sequence
Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

hover_name A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

labels	Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.
branchvalues	"total" or "remainder" Determines how the items in values are summed. Whenset to "total", items in values are taken to be valueof all its descendants. When set to "remainder", itemsin values corresponding to the root and the branches:sectors are taken to be the extra part not part of thesum of the values at their leaves.
maxdepth	Positive integer Sets the number of rendered sectors from any given level. Set maxdepth to -1 to render all thelevels in the hierarchy.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_tips() |> px_icicle(path = ~c(day, time, sex), values = ~total_bill)
```

---

px_imshow *Display an image or a matrix*

Description

Displays a matrix as a heatmap, or an RGB/RGBA array as an image, like plotly.py's px.imshow(). Higher-dimensional arrays can be split into facets and animation frames with facet_col, facet_row and animation_frame.

Usage

```
px_imshow(
  img,
  zmin = NULL,
  zmax = NULL,
  origin = NULL,
  labels = list(),
  x = NULL,
  y = NULL,
  animation_frame = NULL,
  facet_col = NULL,
  facet_row = NULL,
  facet_col_wrap = NULL,
  facet_col_spacing = NULL,
  facet_row_spacing = NULL,
  color_continuous_scale = NULL,
  color_continuous_midpoint = NULL,
  range_color = NULL,
  title = NULL,
  template = NULL,
  width = NULL,
  height = NULL,
  aspect = NULL,
  contrast_rescaling = NULL,
  binary_string = NULL,
  binary_backend = "auto",
  binary_compression_level = 4,
  binary_format = "png",
  text_auto = FALSE
)
```

Arguments

`img`	The image: a numeric or logical matrix (shown as a heatmap with a continuous color scale), a data.frame (its column and row names give `x` and `y`), or an array of dimension `c(height, width, 3)` (RGB) or `c(height, width, 4)` (RGBA) shown as an image. Raw vectors and integer arrays with values 0-255 are treated as 8-bit images; numeric images are scaled according to their range (e.g. 0-1 for `png::readPNG()` output). With `facet_col`, `facet_row` or `animation_frame`, one more dimension per argument.
`zmin`, `zmax`	Minimum and maximum of the color range (per channel for RGB(A) images: a single value, or 1, 3 or 4 values).
`origin`	`"upper"` (default, row 1 at the top) or `"lower"`.
`labels`	Named list overriding the labels of `x`, `y`, `color`, `facet_col`, `facet_row` and `animation_frame`.
`x`, `y`	Coordinates of the columns and rows (numeric for RGB images).

`animation_frame`, `facet_col`, `facet_row`
: Dimension of `img` (a number, or a name in `names(dimnames(img))`) to split into animation frames, facet columns or facet rows.

`facet_col_wrap`
: Maximum number of facet columns.

`facet_col_spacing`, `facet_row_spacing`
: Spacing between facets, as a fraction of the figure.

`color_continuous_scale`, `color_continuous_midpoint`, `range_color`
: Color scale settings for single-channel images (see px_args).

`title`, `template`, `width`, `height`
: See px_args.

`aspect`
: `"equal"` (default for arrays; square pixels) or `"auto"` (fill the figure).

`contrast_rescaling`
: `"minmax"` (scale to the data range; default for single-channel images) or `"infer"` (scale to the range implied by the data type).

`binary_string`
: If `TRUE`, RGB(A) images are sent as a compressed PNG (needs the png package) instead of an array of values; the default when png is installed.

`binary_backend`, `binary_compression_level`
: Unused; accepted for compatibility with plotly.py.

`binary_format`
: Only `"png"` is supported.

`text_auto`
: If `TRUE` or a d3-format string, show the values as text on heatmaps.

Value

A `go_figure()`.

See Also

Other plotly express: `px_area()`, `px_bar()`, `px_bar_polar()`, `px_box()`, `px_choropleth()`, `px_choropleth_map()`, `px_density_contour()`, `px_density_heatmap()`, `px_density_map()`, `px_ecdf()`, `px_funnel()`, `px_funnel_area()`, `px_histogram()`, `px_icicle()`, `px_line()`, `px_line_3d()`, `px_line_geo()`, `px_line_map()`, `px_line_polar()`, `px_line_ternary()`, `px_parallel_categories()`, `px_parallel_coordinates()`, `px_pie()`, `px_scatter()`, `px_scatter_3d()`, `px_scatter_geo()`, `px_scatter_map()`, `px_scatter_matrix()`, `px_scatter_polar()`, `px_scatter_ternary()`, `px_strip()`, `px_sunburst()`, `px_timeline()`, `px_treemap()`, `px_violin()`

Examples

```
px_imshow(matrix(c(1, 20, 30, 20, 1, 60, 30, 60, 1), 3), text_auto = TRUE)

img <- array(c(255, 0, 0, 0, 255, 0, 0, 0, 255, 255, 255, 0), dim = c(2, 2, 3))
px_imshow(img, binary_string = FALSE)

volume <- array(stats::runif(5 * 20 * 20), dim = c(5, 20, 20))
px_imshow(volume, animation_frame = 1)
```

px_line *Line plot*

Description

In a 2D line plot, each row of `data_frame` is represented as a vertex of a polyline mark in 2D space.

Usage

```
px_line(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  line_group = NULL,
  color = NULL,
  line_dash = NULL,
  symbol = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  text = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  error_x = NULL,
  error_x_minus = NULL,
  error_y = NULL,
  error_y_minus = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  orientation = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  line_dash_sequence = NULL,
  line_dash_map = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  markers = FALSE,
  log_x = FALSE,
  log_y = FALSE,
  range_x = NULL,
  range_y = NULL,
  line_shape = NULL,
```

```
  render_mode = "auto",
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
line_group	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to group rows of data_frame into lines.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
line_dash	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign dash-patterns to lines.
symbol	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.
hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.
custom_data	Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

text A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.

facet_row A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.

facet_col A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.

facet_col_wrap Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.

facet_row_spacing
Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

facet_col_spacing
Spacing between facet columns, in paper units Default is 0.02.

error_x A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size x-axis error bars. If error_x_minus is NULL, error bars will be symmetrical, otherwise error_x is used for the positive direction only.

error_x_minus A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size x-axis error bars in the negative direction. Ignored if error_x is NULL.

error_y A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size y-axis error bars. If error_y_minus is NULL, error bars will be symmetrical, otherwise error_y is used for the positive direction only.

error_y_minus A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size y-axis error bars in the negative direction. Ignored if error_y is NULL.

animation_frame
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. `list(day = c("Thur", "Fri", "Sat", "Sun"))`. By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. `c(gdpPercap = "GDP per capita")`.

orientation
: (default `"v"` if x and y are provided and both continuous or both categorical, otherwise `"v"`(`"h"`) if x(y) is categorical and y(x) is continuous, otherwise `"v"`(`"h"`) if only x(y) is provided)

color_discrete_sequence
: Character vector of colors, e.g. `px_colors$qualitative$Set1`. Values of `color` are assigned colors in order (cycling), unless set in `color_discrete_map`. Defaults to the template's `colorway`.

color_discrete_map
: Named character vector or list mapping values of `color` to colors, overriding `color_discrete_sequence`. If the values of `color` are colors, `px_identity()` (or `"identity"`) uses them directly.

line_dash_sequence
: Strings should define valid plotly.js dash-patterns. When `line_dash` is set, values in that column are assigned dash-patterns by cycling through `line_dash_sequence` in the order described in `category_orders`, unless the value of `line_dash` is a key in `line_dash_map`.

line_dash_map
: Named character vector or list mapping values of `line_dash` to dash patterns, overriding `line_dash_sequence`. `px_identity()` uses the values of `line_dash` directly.

symbol_sequence
: Strings should define valid plotly.js symbols. When `symbol` is set, values in that column are assigned symbols by cycling through `symbol_sequence` in the order described in `category_orders`, unless the value of `symbol` is a key in `symbol_map`.

symbol_map
: Named character vector or list mapping values of `symbol` to marker symbols, overriding `symbol_sequence`. `px_identity()` uses the values of `symbol` directly.

markers
: If `TRUE`, markers are shown on lines.

log_x
: If `TRUE`, the x-axis is log-scaled in cartesian coordinates.

log_y
: If `TRUE`, the y-axis is log-scaled in cartesian coordinates.

range_x
: If provided, overrides auto-scaling on the x-axis in cartesian coordinates.

range_y
: If provided, overrides auto-scaling on the y-axis in cartesian coordinates.

line_shape
: One of `"linear"`, `"spline"`, `"hv"`, `"vh"`, `"hvh"`, or `"vhv"`

render_mode
: One of `"auto"`, `"svg"` or `"webgl"`, default `"auto"` Controls the browser API used to draw marks. `"svg"` is appropriate for figures of less than 1000 data points, and will allow for fully-vectorized output. `"webgl"` is likely necessary for acceptable performance above 1000 points but rasterizes part of the output. `"auto"` uses heuristics to choose the mode.

title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_gapminder()[continent == "Oceania"] |>
  px_line(x = ~year, y = ~lifeExp, color = ~country, markers = TRUE)
```

---

px_line_3d *3D line plot*

Description

In a 3D line plot, each row of data_frame is represented as a vertex of a polyline mark in 3D space.

Usage

```
px_line_3d(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  z = NULL,
  color = NULL,
  line_dash = NULL,
  text = NULL,
  line_group = NULL,
  symbol = NULL,
  hover_name = NULL,
  hover_data = NULL,
```

```
  custom_data = NULL,
  error_x = NULL,
  error_x_minus = NULL,
  error_y = NULL,
  error_y_minus = NULL,
  error_z = NULL,
  error_z_minus = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  line_dash_sequence = NULL,
  line_dash_map = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  markers = FALSE,
  log_x = FALSE,
  log_y = FALSE,
  log_z = FALSE,
  range_x = NULL,
  range_y = NULL,
  range_z = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
z	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the z axis in cartesian coordinates.

color — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.

line_dash — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign dash-patterns to lines.

text — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.

line_group — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to group rows of data_frame into lines.

symbol — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.

hover_name — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data — Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data — Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

error_x — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size x-axis error bars. If error_x_minus is NULL, error bars will be symmetrical, otherwise error_x is used for the positive direction only.

error_x_minus — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size x-axis error bars in the negative direction. Ignored if error_x is NULL.

error_y — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size y-axis error bars. If error_y_minus is NULL, error bars will be symmetrical, otherwise error_y is used for the positive direction only.

error_y_minus — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size y-axis error bars in the negative direction. Ignored if error_y is NULL.

error_z A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size z-axis error bars. If error_z_minus is NULL, error bars will be symmetrical, otherwise error_z is used for the positive direction only.

error_z_minus A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size z-axis error bars in the negative direction. Ignored if error_z is NULL.

animation_frame
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

line_dash_sequence
Strings should define valid plotly.js dash-patterns. When line_dash is set, values in that column are assigned dash-patterns by cycling through line_dash_sequence in the order described in category_orders, unless the value of line_dash is a key in line_dash_map.

line_dash_map Named character vector or list mapping values of line_dash to dash patterns, overriding line_dash_sequence. px_identity() uses the values of line_dash directly.

symbol_sequence
Strings should define valid plotly.js symbols. When symbol is set, values in that column are assigned symbols by cycling through symbol_sequence in the order described in category_orders, unless the value of symbol is a key in symbol_map.

symbol_map	Named character vector or list mapping values of symbol to marker symbols, overriding symbol_sequence. px_identity() uses the values of symbol directly.
markers	If TRUE, markers are shown on lines.
log_x	If TRUE, the x-axis is log-scaled in cartesian coordinates.
log_y	If TRUE, the y-axis is log-scaled in cartesian coordinates.
log_z	If TRUE, the z-axis is log-scaled in cartesian coordinates.
range_x	If provided, overrides auto-scaling on the x-axis in cartesian coordinates.
range_y	If provided, overrides auto-scaling on the y-axis in cartesian coordinates.
range_z	If provided, overrides auto-scaling on the z-axis in cartesian coordinates.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_gapminder()[continent == "Europe"] |>
  px_line_3d(x = ~gdpPercap, y = ~pop, z = ~year, color = ~country)
```

---

px_line_geo *Line plot on a geographic map*

Description

In a geographic line plot, each row of data_frame is represented as a vertex of a polyline mark on a map.

Usage

```
px_line_geo(
  data_frame = NULL,
  lat = NULL,
  lon = NULL,
  locations = NULL,
  locationmode = NULL,
  geojson = NULL,
  featureidkey = NULL,
  color = NULL,
  line_dash = NULL,
  text = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  line_group = NULL,
  symbol = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  line_dash_sequence = NULL,
  line_dash_map = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  markers = FALSE,
  projection = NULL,
  scope = NULL,
  center = NULL,
  fitbounds = NULL,
  basemap_visible = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
lat	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks according to latitude on a map.
lon	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks according to longitude on a map.
locations	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are to be interpreted according to locationmode and mapped to longitude/latitude.
locationmode	One of "ISO-3", "USA-states", or 'country names' Determines the set of locations used to match entries in locations to regions on the map.
geojson	GeoJSON as a list (e.g. from jsonlite::read_json()), with features whose ids are referenced by locations.
featureidkey	Path to field in GeoJSON feature object with which to match the values passed in to locations.The most common alternative to the default is of the form 'properties.<key>.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
line_dash	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign dash-patterns to lines.
text	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.
facet_row	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.
facet_col	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.
facet_col_wrap	Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.

facet_row_spacing
: Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

facet_col_spacing
: Spacing between facet columns, in paper units Default is 0.02.

hover_name
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data
: Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data
: Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

line_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to group rows of data_frame into lines.

symbol
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.

animation_frame
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

line_dash_sequence
: Strings should define valid plotly.js dash-patterns. When line_dash is set, values in that column are assigned dash-patterns by cycling through line_dash_sequence in the order described in category_orders, unless the value of line_dash is a key in line_dash_map.

line_dash_map
: Named character vector or list mapping values of line_dash to dash patterns, overriding line_dash_sequence. px_identity() uses the values of line_dash directly.

symbol_sequence
: Strings should define valid plotly.js symbols. When symbol is set, values in that column are assigned symbols by cycling through symbol_sequence in the order described in category_orders, unless the value of symbol is a key in symbol_map.

symbol_map
: Named character vector or list mapping values of symbol to marker symbols, overriding symbol_sequence. px_identity() uses the values of symbol directly.

markers
: If TRUE, markers are shown on lines.

projection
: One of "equirectangular", "mercator", "orthographic", "natural earth", "kavrayskiy7", "miller", "robinson", "eckert4", "azimuthal equal area", "azimuthal equidistant", "conic equal area", "conic conformal", "conic equidistant", "gnomonic", "stereographic", "mollweide", "hammer", "transverse mercator", "albers usa", "winkel tripel", "aitoff", or "sinusoidal"Default depends on scope.

scope
: One of "world", "usa", "europe", "asia", "africa", "north america", or "south america"Default is "world" unless projection is set to "albers usa", which forces "usa".

center
: Dict keys are "lat" and "lon" Sets the center point of the map.

fitbounds
: One of FALSE, locations or geojson.

basemap_visible
: Force the basemap visibility.

title
: The figure title.

subtitle
: The figure subtitle.

template
: A template name (see template_names()) or a template list. Defaults to default_template().

width
: The figure width in pixels.

height
: The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_gapminder(year = 2007)[continent == "Oceania"] |>
  px_line_geo(locations = ~iso_alpha)
```

---

px_line_map *Line plot on a tile map*

Description

In a line map, each row of `data_frame` is represented as a vertex of a polyline mark on the map.

Usage

```
px_line_map(
  data_frame = NULL,
  lat = NULL,
  lon = NULL,
  color = NULL,
  text = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  line_group = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  zoom = NULL,
  center = NULL,
  map_style = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
lat	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks according to latitude on a map.
lon	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks according to longitude on a map.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
text	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.
hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.
custom_data	Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.
line_group	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to group rows of data_frame into lines.
animation_frame	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.
animation_group	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. `list(day = c("Thur", "Fri", "Sat", "Sun"))`. By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. `c(gdpPercap = "GDP per capita")`.

color_discrete_sequence
: Character vector of colors, e.g. `px_colors$qualitative$Set1`. Values of `color` are assigned colors in order (cycling), unless set in `color_discrete_map`. Defaults to the template's `colorway`.

color_discrete_map
: Named character vector or list mapping values of `color` to colors, overriding `color_discrete_sequence`. If the values of `color` are colors, `px_identity()` (or `"identity"`) uses them directly.

zoom
: Between 0 and 20. Sets initial map zoom level. Higher values are more zoomed-in. If not provided, the map will select its own zoom level.

center
: Dict keys are `"lat"` and `"lon"` Sets the center point of the map.

map_style
: Identifier of base map style. Allowed values are `"basic"`, `"carto-darkmatter"`, `"carto-darkmatter-nolabels"`, `"carto-positron"`, `"carto-positron-nolabels"`, `"carto-voyager"`, `"carto-voyager-nolabels"`, `"dark"`, `"light"`, `"open-street-map"`, `"outdoors"`, `"satellite"`, `"satellite-streets"`, `"streets"`, `"white-bg"`.

title
: The figure title.

subtitle
: The figure subtitle.

template
: A template name (see `template_names()`) or a template list. Defaults to `default_template()`.

width
: The figure width in pixels.

height
: The figure height in pixels.

Value

A `go_figure()`.

See Also

Other plotly express: `px_area()`, `px_bar()`, `px_bar_polar()`, `px_box()`, `px_choropleth()`, `px_choropleth_map()`, `px_density_contour()`, `px_density_heatmap()`, `px_density_map()`, `px_ecdf()`, `px_funnel()`, `px_funnel_area()`, `px_histogram()`, `px_icicle()`, `px_imshow()`, `px_line()`, `px_line_3d()`, `px_line_geo()`, `px_line_polar()`, `px_line_ternary()`, `px_parallel_categories()`, `px_parallel_coordinates()`, `px_pie()`, `px_scatter()`, `px_scatter_3d()`, `px_scatter_geo()`, `px_scatter_map()`, `px_scatter_matrix()`, `px_scatter_polar()`, `px_scatter_ternary()`, `px_strip()`, `px_sunburst()`, `px_timeline()`, `px_treemap()`, `px_violin()`

Examples

```
px_data_carshare()[1:20] |>
  px_line_map(lat = ~centroid_lat, lon = ~centroid_lon, zoom = 10)
```

px_line_polar *Polar line plot*

Description

In a polar line plot, each row of `data_frame` is represented as a vertex of a polyline mark in polar coordinates.

Usage

```
px_line_polar(
  data_frame = NULL,
  r = NULL,
  theta = NULL,
  color = NULL,
  line_dash = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  line_group = NULL,
  text = NULL,
  symbol = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  line_dash_sequence = NULL,
  line_dash_map = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  markers = FALSE,
  direction = "clockwise",
  start_angle = 90,
  line_close = FALSE,
  line_shape = NULL,
  render_mode = "auto",
  range_r = NULL,
  range_theta = NULL,
  log_r = FALSE,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
r	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the radial axis in polar coordinates.
theta	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the angular axis in polar coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
line_dash	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign dash-patterns to lines.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.
hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.
custom_data	Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.
line_group	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to group rows of data_frame into lines.
text	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.
symbol	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.
animation_frame	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

line_dash_sequence
: Strings should define valid plotly.js dash-patterns. When line_dash is set, values in that column are assigned dash-patterns by cycling through line_dash_sequence in the order described in category_orders, unless the value of line_dash is a key in line_dash_map.

line_dash_map
: Named character vector or list mapping values of line_dash to dash patterns, overriding line_dash_sequence. px_identity() uses the values of line_dash directly.

symbol_sequence
: Strings should define valid plotly.js symbols. When symbol is set, values in that column are assigned symbols by cycling through symbol_sequence in the order described in category_orders, unless the value of symbol is a key in symbol_map.

symbol_map
: Named character vector or list mapping values of symbol to marker symbols, overriding symbol_sequence. px_identity() uses the values of symbol directly.

markers
: If TRUE, markers are shown on lines.

direction
: One of "counterclockwise" or "clockwise". Default is "clockwise" Sets the direction in which increasing values of the angular axis are drawn.

start_angle
: Sets start angle for the angular axis, with 0 being due east and 90 being due north.

line_close
: If TRUE, an extra line segment is drawn between the first and last point.

line_shape
: One of "linear", "spline", "hv", "vh", "hvh", or "vhv"

render_mode	One of "auto", "svg" or "webgl", default "auto" Controls the browser API used to draw marks. "svg" is appropriate for figures of less than 1000 data points, and will allow for fully-vectorized output. "webgl" is likely necessary for acceptable performance above 1000 points but rasterizes part of the output. "auto" uses heuristics to choose the mode.
range_r	If provided, overrides auto-scaling on the radial axis in polar coordinates.
range_theta	If provided, overrides auto-scaling on the angular axis in polar coordinates.
log_r	If TRUE, the radial axis is log-scaled in polar coordinates.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_wind() |>
  px_line_polar(r = ~frequency, theta = ~direction, color = ~strength, line_close = TRUE)
```

---

px_line_ternary *Ternary line plot*

Description

In a ternary line plot, each row of data_frame is represented as a vertex of a polyline mark in ternary coordinates.

Usage

```
px_line_ternary(
  data_frame = NULL,
  a = NULL,
  b = NULL,
  c = NULL,
  color = NULL,
  line_dash = NULL,
  line_group = NULL,
  symbol = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  text = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  line_dash_sequence = NULL,
  line_dash_map = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  markers = FALSE,
  line_shape = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
a	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the a axis in ternary coordinates.
b	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the b axis in ternary coordinates.
c	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values

from this column or vector are used to position marks along the c axis in ternary coordinates.

`color` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to assign color to marks.

`line_dash` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to assign dash-patterns to lines.

`line_group` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to group rows of `data_frame` into lines.

`symbol` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to assign symbols to marks.

`hover_name` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector appear in bold in the hover tooltip.

`hover_data` Extra data to show in the hover tooltip: column names (a character vector or a formula such as `~c(a, b)`), or a named list with column names as names and values `TRUE` (default formatting), `FALSE` (hide the column, e.g. one shown by default), a d3-format string such as `":.3f"`, a vector of data, or a list of a format string or `TRUE` and a vector of data.

`custom_data` Column names (a character vector or a formula such as `~c(a, b)`), or a list of column names, formulas and vectors. Values from these columns are added to the traces' `customdata` (e.g. for use in Shiny events) without appearing in the hover tooltip.

`text` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector appear in the figure as text labels.

`animation_frame`
A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to assign marks to animation frames.

`animation_group`
A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to provide object-constancy across animation frames: rows with matching `animation_groups` will be treated as if they describe the same object in each frame.

`category_orders`
Named list mapping column names to the order of their values in axes, legends and facets, e.g. `list(day = c("Thur", "Fri", "Sat", "Sun"))`. By default values appear in the order first encountered, or in level order for factors.

`labels` Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. `c(gdpPercap = "GDP per capita")`.

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

line_dash_sequence
: Strings should define valid plotly.js dash-patterns. When line_dash is set, values in that column are assigned dash-patterns by cycling through line_dash_sequence in the order described in category_orders, unless the value of line_dash is a key in line_dash_map.

line_dash_map
: Named character vector or list mapping values of line_dash to dash patterns, overriding line_dash_sequence. px_identity() uses the values of line_dash directly.

symbol_sequence
: Strings should define valid plotly.js symbols. When symbol is set, values in that column are assigned symbols by cycling through symbol_sequence in the order described in category_orders, unless the value of symbol is a key in symbol_map.

symbol_map
: Named character vector or list mapping values of symbol to marker symbols, overriding symbol_sequence. px_identity() uses the values of symbol directly.

markers
: If TRUE, markers are shown on lines.

line_shape
: One of "linear", "spline", "hv", "vh", "hvh", or "vhv"

title
: The figure title.

subtitle
: The figure subtitle.

template
: A template name (see template_names()) or a template list. Defaults to default_template().

width
: The figure width in pixels.

height
: The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_election()[1:5] |> px_line_ternary(a = ~Joly, b = ~Coderre, c = ~Bergeron)
```

`px_parallel_categories` *Parallel categories plot*

Description

In a parallel categories (or parallel sets) plot, each row of `data_frame` is grouped with other rows that share the same values of `dimensions` and then plotted as a polyline mark through a set of parallel axes, one for each of the `dimensions`.

Usage

```
px_parallel_categories(
  data_frame = NULL,
  dimensions = NULL,
  color = NULL,
  labels = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL,
  dimensions_max_cardinality = 50
)
```

Arguments

`data_frame`	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if `NULL`, a data frame is built from the vectors given in the other arguments.
`dimensions`	Column names in `data_frame` (a character vector or a formula such as `~c(a, b)`), or a list of column names, formulas and vectors. Values from these columns are used for multidimensional visualization.
`color`	A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to assign color to marks.
`labels`	Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. `c(gdpPercap = "GDP per capita")`.

color_continuous_scale
: Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color
: If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
: If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

title
: The figure title.

subtitle
: The figure subtitle.

template
: A template name (see template_names()) or a template list. Defaults to default_template().

width
: The figure width in pixels.

height
: The figure height in pixels.

dimensions_max_cardinality
: When dimensions is NULL and data_frame is provided, columns with more than this number of unique values are excluded from the output. Not used when dimensions is passed.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_tips() |> px_parallel_categories(dimensions = ~c(sex, smoker, day), color = ~size)
```

---

px_parallel_coordinates

Parallel coordinates plot

Description

In a parallel coordinates plot, each row of data_frame is represented by a polyline mark which traverses a set of parallel axes, one for each of the dimensions.

Usage

```
px_parallel_coordinates(
  data_frame = NULL,
  dimensions = NULL,
  color = NULL,
  labels = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
dimensions	Column names in data_frame (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are used for multidimensional visualization.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
labels	Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").
color_continuous_scale	Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.
range_color	If provided, overrides auto-scaling on the continuous color scale.
color_continuous_midpoint	If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_iris() |>
  px_parallel_coordinates(color = ~species_id,
                  dimensions = ~c(sepal_width, sepal_length, petal_width, petal_length))
```

---

px_pie *Pie chart*

Description

In a pie plot, each row of data_frame is represented as a sector of a pie.

Usage

```
px_pie(
  data_frame = NULL,
  names = NULL,
  values = NULL,
  color = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  category_orders = NULL,
  labels = NULL,
  title = NULL,
  subtitle = NULL,
```

```
  template = NULL,
  width = NULL,
  height = NULL,
  opacity = NULL,
  hole = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
names	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used as labels for sectors.
values	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to set values associated to sectors.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
facet_row	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.
facet_col	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.
facet_col_wrap	Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.
facet_row_spacing	Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.
facet_col_spacing	Spacing between facet columns, in paper units Default is 0.02.
color_discrete_sequence	Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.
color_discrete_map	Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

hover_name — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data — Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data — Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

category_orders — Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels — Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

title — The figure title.

subtitle — The figure subtitle.

template — A template name (see template_names()) or a template list. Defaults to default_template().

width — The figure width in pixels.

height — The figure height in pixels.

opacity — Value between 0 and 1. Sets the opacity for markers.

hole — Number between 0 and 1: the fraction of the radius cut out of the pie to make a donut chart.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_tips() |> px_pie(values = ~tip, names = ~day, hole = 0.3)
```

px_scatter *Scatter plot*

Description

In a scatter plot, each row of `data_frame` is represented by a symbol mark in 2D space.

Usage

```
px_scatter(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  color = NULL,
  symbol = NULL,
  size = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  text = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  error_x = NULL,
  error_x_minus = NULL,
  error_y = NULL,
  error_y_minus = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  orientation = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  opacity = NULL,
  size_max = NULL,
  marginal_x = NULL,
  marginal_y = NULL,
  trendline = NULL,
  trendline_options = NULL,
```

```
  trendline_color_override = NULL,
  trendline_scope = "trace",
  log_x = FALSE,
  log_y = FALSE,
  range_x = NULL,
  range_y = NULL,
  render_mode = "auto",
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
symbol	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.
size	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign mark sizes.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.
hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.
custom_data	Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to

the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

text A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.

facet_row A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.

facet_col A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.

facet_col_wrap Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.

facet_row_spacing
Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

facet_col_spacing
Spacing between facet columns, in paper units Default is 0.02.

error_x A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size x-axis error bars. If error_x_minus is NULL, error bars will be symmetrical, otherwise error_x is used for the positive direction only.

error_x_minus A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size x-axis error bars in the negative direction. Ignored if error_x is NULL.

error_y A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size y-axis error bars. If error_y_minus is NULL, error bars will be symmetrical, otherwise error_y is used for the positive direction only.

error_y_minus A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size y-axis error bars in the negative direction. Ignored if error_y is NULL.

animation_frame
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values

from this column or vector are used to provide object-constancy across animation frames: rows with matching `animation_groups` will be treated as if they describe the same object in each frame.

`category_orders`
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. `list(day = c("Thur", "Fri", "Sat", "Sun"))`. By default values appear in the order first encountered, or in level order for factors.

`labels`
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. `c(gdpPercap = "GDP per capita")`.

`orientation`
: (default `"v"` if `x` and `y` are provided and both continuous or both categorical, otherwise `"v"`(`"h"`) if `x`(`y`) is categorical and `y`(`x`) is continuous, otherwise `"v"`(`"h"`) if only `x`(`y`) is provided)

`color_discrete_sequence`
: Character vector of colors, e.g. `px_colors$qualitative$Set1`. Values of `color` are assigned colors in order (cycling), unless set in `color_discrete_map`. Defaults to the template's `colorway`.

`color_discrete_map`
: Named character vector or list mapping values of `color` to colors, overriding `color_discrete_sequence`. If the values of `color` are colors, `px_identity()` (or `"identity"`) uses them directly.

`color_continuous_scale`
: Character vector of colors, e.g. `px_colors$sequential$Viridis`, or the name of a scale in px_colors. Used when `color` is numeric. Defaults to the template's sequential colorscale.

`range_color`
: If provided, overrides auto-scaling on the continuous color scale.

`color_continuous_midpoint`
: If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using `plotly.express.colors.diverging` color scales as the inputs to `color_continuous_scale`.

`symbol_sequence`
: Strings should define valid plotly.js symbols. When `symbol` is set, values in that column are assigned symbols by cycling through `symbol_sequence` in the order described in `category_orders`, unless the value of `symbol` is a key in `symbol_map`.

`symbol_map`
: Named character vector or list mapping values of `symbol` to marker symbols, overriding `symbol_sequence`. `px_identity()` uses the values of `symbol` directly.

`opacity`
: Value between 0 and 1. Sets the opacity for markers.

`size_max`
: Set the maximum mark size when using `size`.

`marginal_x`
: One of `"rug"`, `"box"`, `"violin"`, or `"histogram"`. If set, a horizontal subplot is drawn above the main plot, visualizing the x-distribution.

`marginal_y`
: One of `"rug"`, `"box"`, `"violin"`, or `"histogram"`. If set, a vertical subplot is drawn to the right of the main plot, visualizing the y-distribution.

trendline — One of "ols" (ordinary least squares regression), "lowess" (locally weighted smoothing), "rolling", "expanding" or "ewm" (exponentially weighted moving average). A trendline is drawn for each discrete color/symbol group; configure it with trendline_options. Fitted OLS models are available from px_get_trendline_results().

trendline_options — Named list of options for the trendline function: for "ols", add_constant, log_x and log_y; for "lowess", frac; for "rolling", window, min_periods, center and function ("mean", "median", "sum", "min", "max", "std", "var", "count"); for "expanding", min_periods and function; for "ewm", one of com, span, halflife or alpha, and adjust.

trendline_color_override — Valid CSS color. If provided, and if trendline is set, all trendlines will be drawn in this color rather than in the same color as the traces from which they draw their inputs.

trendline_scope — If "trace", then one trendline is drawn per trace (i.e. per color, symbol, facet, animation frame etc) and if "overall" then one trendline is computed for the entire dataset, and replicated across all facets.

log_x — If TRUE, the x-axis is log-scaled in cartesian coordinates.

log_y — If TRUE, the y-axis is log-scaled in cartesian coordinates.

range_x — If provided, overrides auto-scaling on the x-axis in cartesian coordinates.

range_y — If provided, overrides auto-scaling on the y-axis in cartesian coordinates.

render_mode — One of "auto", "svg" or "webgl", default "auto" Controls the browser API used to draw marks. "svg" is appropriate for figures of less than 1000 data points, and will allow for fully-vectorized output. "webgl" is likely necessary for acceptable performance above 1000 points but rasterizes part of the output. "auto" uses heuristics to choose the mode.

title — The figure title.

subtitle — The figure subtitle.

template — A template name (see template_names()) or a template list. Defaults to default_template().

width — The figure width in pixels.

height — The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_iris() |>
  px_scatter(x = ~sepal_width, y = ~sepal_length, color = ~species,
             size = ~petal_length, hover_data = ~petal_width)
px_data_tips() |> px_scatter(x = ~total_bill, y = ~tip, facet_col = ~time, trendline = "ols")
```

`px_scatter_3d` *3D scatter plot*

Description

In a 3D scatter plot, each row of `data_frame` is represented by a symbol mark in 3D space.

Usage

```
px_scatter_3d(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  z = NULL,
  color = NULL,
  symbol = NULL,
  size = NULL,
  text = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  error_x = NULL,
  error_x_minus = NULL,
  error_y = NULL,
  error_y_minus = NULL,
  error_z = NULL,
  error_z_minus = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  size_max = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  opacity = NULL,
  log_x = FALSE,
```

```
  log_y = FALSE,
  log_z = FALSE,
  range_x = NULL,
  range_y = NULL,
  range_z = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
z	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the z axis in cartesian coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
symbol	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.
size	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign mark sizes.
text	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.
hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and

values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

error_x A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size x-axis error bars. If error_x_minus is NULL, error bars will be symmetrical, otherwise error_x is used for the positive direction only.

error_x_minus A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size x-axis error bars in the negative direction. Ignored if error_x is NULL.

error_y A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size y-axis error bars. If error_y_minus is NULL, error bars will be symmetrical, otherwise error_y is used for the positive direction only.

error_y_minus A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size y-axis error bars in the negative direction. Ignored if error_y is NULL.

error_z A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size z-axis error bars. If error_z_minus is NULL, error bars will be symmetrical, otherwise error_z is used for the positive direction only.

error_z_minus A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to size z-axis error bars in the negative direction. Ignored if error_z is NULL.

animation_frame
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
Named list mapping column names to the order of their values in axes, legends

and facets, e.g. `list(day = c("Thur", "Fri", "Sat", "Sun"))`. By default values appear in the order first encountered, or in level order for factors.

`labels` Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. `c(gdpPercap = "GDP per capita")`.

`size_max` Set the maximum mark size when using `size`.

`color_discrete_sequence`
Character vector of colors, e.g. `px_colors$qualitative$Set1`. Values of `color` are assigned colors in order (cycling), unless set in `color_discrete_map`. Defaults to the template's `colorway`.

`color_discrete_map`
Named character vector or list mapping values of `color` to colors, overriding `color_discrete_sequence`. If the values of `color` are colors, `px_identity()` (or `"identity"`) uses them directly.

`color_continuous_scale`
Character vector of colors, e.g. `px_colors$sequential$Viridis`, or the name of a scale in px_colors. Used when `color` is numeric. Defaults to the template's sequential colorscale.

`range_color` If provided, overrides auto-scaling on the continuous color scale.

`color_continuous_midpoint`
If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using `plotly.express.colors.diverging` color scales as the inputs to `color_continuous_scale`.

`symbol_sequence`
Strings should define valid plotly.js symbols. When `symbol` is set, values in that column are assigned symbols by cycling through `symbol_sequence` in the order described in `category_orders`, unless the value of `symbol` is a key in `symbol_map`.

`symbol_map` Named character vector or list mapping values of `symbol` to marker symbols, overriding `symbol_sequence`. `px_identity()` uses the values of `symbol` directly.

`opacity` Value between 0 and 1. Sets the opacity for markers.

`log_x` If `TRUE`, the x-axis is log-scaled in cartesian coordinates.

`log_y` If `TRUE`, the y-axis is log-scaled in cartesian coordinates.

`log_z` If `TRUE`, the z-axis is log-scaled in cartesian coordinates.

`range_x` If provided, overrides auto-scaling on the x-axis in cartesian coordinates.

`range_y` If provided, overrides auto-scaling on the y-axis in cartesian coordinates.

`range_z` If provided, overrides auto-scaling on the z-axis in cartesian coordinates.

`title` The figure title.

`subtitle` The figure subtitle.

`template` A template name (see `template_names()`) or a template list. Defaults to `default_template()`.

`width` The figure width in pixels.

`height` The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_iris() |>
  px_scatter_3d(x = ~sepal_length, y = ~sepal_width, z = ~petal_width, color = ~species)
```

---

px_scatter_geo *Scatter plot on a geographic map*

Description

In a geographic scatter plot, each row of data_frame is represented by a symbol mark on a map.

Usage

```
px_scatter_geo(
  data_frame = NULL,
  lat = NULL,
  lon = NULL,
  locations = NULL,
  locationmode = NULL,
  geojson = NULL,
  featureidkey = NULL,
  color = NULL,
  text = NULL,
  symbol = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  size = NULL,
```

```
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  opacity = NULL,
  size_max = NULL,
  projection = NULL,
  scope = NULL,
  center = NULL,
  fitbounds = NULL,
  basemap_visible = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
lat	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks according to latitude on a map.
lon	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks according to longitude on a map.
locations	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are to be interpreted according to locationmode and mapped to longitude/latitude.
locationmode	One of "ISO-3", "USA-states", or 'country names' Determines the set of locations used to match entries in locations to regions on the map.
geojson	GeoJSON as a list (e.g. from jsonlite::read_json()), with features whose ids are referenced by locations.

featureidkey Path to field in GeoJSON feature object with which to match the values passed in to locations.The most common alternative to the default is of the form 'properties.<key>.

color A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.

text A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.

symbol A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.

facet_row A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.

facet_col A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.

facet_col_wrap Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.

facet_row_spacing
Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

facet_col_spacing
Spacing between facet columns, in paper units Default is 0.02.

hover_name A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

size A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign mark sizes.

animation_frame
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

color_continuous_scale
: Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color
: If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
: If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

symbol_sequence
: Strings should define valid plotly.js symbols. When symbol is set, values in that column are assigned symbols by cycling through symbol_sequence in the order described in category_orders, unless the value of symbol is a key in symbol_map.

symbol_map
: Named character vector or list mapping values of symbol to marker symbols, overriding symbol_sequence. px_identity() uses the values of symbol directly.

opacity
: Value between 0 and 1. Sets the opacity for markers.

size_max
: Set the maximum mark size when using size.

projection
: One of "equirectangular", "mercator", "orthographic", "natural earth", "kavrayskiy7", "miller", "robinson", "eckert4", "azimuthal equal area", "azimuthal equidistant", "conic equal area", "conic conformal", "conic equidistant", "gnomonic", "stereographic", "mollweide", "hammer", "transverse mercator", "albers usa", "winkel tripel", "aitoff", or "sinusoidal"Default depends on scope.

scope	One of "world", "usa", "europe", "asia", "africa", "north america", or "south america"Default is "world" unless projection is set to "albers usa", which forces "usa".
center	Dict keys are "lat" and "lon" Sets the center point of the map.
fitbounds	One of FALSE, locations or geojson.
basemap_visible	Force the basemap visibility.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_gapminder(year = 2007) |>
  px_scatter_geo(locations = ~iso_alpha, size = ~pop, color = ~continent)
```

---

px_scatter_map *Scatter plot on a tile map*

Description

In a scatter map, each row of data_frame is represented by a symbol mark on the map.

Usage

```
px_scatter_map(
  data_frame = NULL,
  lat = NULL,
  lon = NULL,
  color = NULL,
  text = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  size = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  opacity = NULL,
  size_max = NULL,
  zoom = NULL,
  center = NULL,
  map_style = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
lat	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks according to latitude on a map.
lon	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks according to longitude on a map.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.

text
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.

hover_name
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data
: Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data
: Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

size
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign mark sizes.

animation_frame
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

color_continuous_scale
: Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using `plotly.express.colors.diverging` color scales as the inputs to `color_continuous_scale`.

opacity Value between 0 and 1. Sets the opacity for markers.

size_max Set the maximum mark size when using `size`.

zoom Between 0 and 20. Sets initial map zoom level. Higher values are more zoomed-in. If not provided, the map will select its own zoom level.

center Dict keys are `"lat"` and `"lon"` Sets the center point of the map.

map_style Identifier of base map style. Allowed values are `"basic"`, `"carto-darkmatter"`, `"carto-darkmatter-nolabels"`, `"carto-positron"`, `"carto-positron-nolabels"`, `"carto-voyager"`, `"carto-voyager-nolabels"`, `"dark"`, `"light"`, `"open-street-map"`, `"outdoors"`, `"satellite"`, `"satellite-streets"`, `"streets"`, `"white-bg"`.

title The figure title.

subtitle The figure subtitle.

template A template name (see `template_names()`) or a template list. Defaults to `default_template()`.

width The figure width in pixels.

height The figure height in pixels.

Value

A `go_figure()`.

See Also

Other plotly express: `px_area()`, `px_bar()`, `px_bar_polar()`, `px_box()`, `px_choropleth()`, `px_choropleth_map()`, `px_density_contour()`, `px_density_heatmap()`, `px_density_map()`, `px_ecdf()`, `px_funnel()`, `px_funnel_area()`, `px_histogram()`, `px_icicle()`, `px_imshow()`, `px_line()`, `px_line_3d()`, `px_line_geo()`, `px_line_map()`, `px_line_polar()`, `px_line_ternary()`, `px_parallel_categories()`, `px_parallel_coordinates()`, `px_pie()`, `px_scatter()`, `px_scatter_3d()`, `px_scatter_geo()`, `px_scatter_matrix()`, `px_scatter_polar()`, `px_scatter_ternary()`, `px_strip()`, `px_sunburst()`, `px_timeline()`, `px_treemap()`, `px_violin()`

Examples

```
px_data_carshare() |>
  px_scatter_map(lat = ~centroid_lat, lon = ~centroid_lon, color = ~peak_hour,
                 size = ~car_hours, zoom = 10)
```

px_scatter_matrix *Scatter plot matrix*

Description

In a scatter plot matrix (or SPLOM), each row of `data_frame` is represented by a multiple symbol marks, one in each cell of a grid of 2D scatter plots, which plot each pair of `dimensions` against each other.

Usage

```
px_scatter_matrix(
  data_frame = NULL,
  dimensions = NULL,
  color = NULL,
  symbol = NULL,
  size = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  opacity = NULL,
  size_max = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

`data_frame`	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if `NULL`, a data frame is built from the vectors given in the other arguments.
`dimensions`	Column names in `data_frame` (a character vector or a formula such as `~c(a, b)`), or a list of column names, formulas and vectors. Values from these columns are used for multidimensional visualization.

color — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.

symbol — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.

size — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign mark sizes.

hover_name — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data — Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data — Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

category_orders — Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels — Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence — Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map — Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

color_continuous_scale — Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color — If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint — If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

`symbol_sequence`
: Strings should define valid plotly.js symbols. When `symbol` is set, values in that column are assigned symbols by cycling through `symbol_sequence` in the order described in `category_orders`, unless the value of `symbol` is a key in `symbol_map`.

`symbol_map`
: Named character vector or list mapping values of `symbol` to marker symbols, overriding `symbol_sequence`. `px_identity()` uses the values of `symbol` directly.

`opacity`
: Value between 0 and 1. Sets the opacity for markers.

`size_max`
: Set the maximum mark size when using `size`.

`title`
: The figure title.

`subtitle`
: The figure subtitle.

`template`
: A template name (see `template_names()`) or a template list. Defaults to `default_template()`.

`width`
: The figure width in pixels.

`height`
: The figure height in pixels.

Value

A `go_figure()`.

See Also

Other plotly express: `px_area()`, `px_bar()`, `px_bar_polar()`, `px_box()`, `px_choropleth()`, `px_choropleth_map()`, `px_density_contour()`, `px_density_heatmap()`, `px_density_map()`, `px_ecdf()`, `px_funnel()`, `px_funnel_area()`, `px_histogram()`, `px_icicle()`, `px_imshow()`, `px_line()`, `px_line_3d()`, `px_line_geo()`, `px_line_map()`, `px_line_polar()`, `px_line_ternary()`, `px_parallel_categories()`, `px_parallel_coordinates()`, `px_pie()`, `px_scatter()`, `px_scatter_3d()`, `px_scatter_geo()`, `px_scatter_map()`, `px_scatter_polar()`, `px_scatter_ternary()`, `px_strip()`, `px_sunburst()`, `px_timeline()`, `px_treemap()`, `px_violin()`

Examples

```
px_data_iris() |>
  px_scatter_matrix(dimensions = ~c(sepal_width, sepal_length, petal_width), color = ~species)
```

---

px_scatter_polar *Polar scatter plot*

Description

In a polar scatter plot, each row of `data_frame` is represented by a symbol mark in polar coordinates.

Usage

```
px_scatter_polar(
  data_frame = NULL,
  r = NULL,
  theta = NULL,
  color = NULL,
  symbol = NULL,
  size = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  text = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  opacity = NULL,
  direction = "clockwise",
  start_angle = 90,
  size_max = NULL,
  range_r = NULL,
  range_theta = NULL,
  log_r = FALSE,
  render_mode = "auto",
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
r	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the radial axis in polar coordinates.

theta — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the angular axis in polar coordinates.

color — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.

symbol — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.

size — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign mark sizes.

hover_name — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data — Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data — Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

text — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.

animation_frame — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group — A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders — Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels — Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

color_continuous_scale
: Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color
: If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
: If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

symbol_sequence
: Strings should define valid plotly.js symbols. When symbol is set, values in that column are assigned symbols by cycling through symbol_sequence in the order described in category_orders, unless the value of symbol is a key in symbol_map.

symbol_map
: Named character vector or list mapping values of symbol to marker symbols, overriding symbol_sequence. px_identity() uses the values of symbol directly.

opacity
: Value between 0 and 1. Sets the opacity for markers.

direction
: One of "counterclockwise" or "clockwise". Default is "clockwise" Sets the direction in which increasing values of the angular axis are drawn.

start_angle
: Sets start angle for the angular axis, with 0 being due east and 90 being due north.

size_max
: Set the maximum mark size when using size.

range_r
: If provided, overrides auto-scaling on the radial axis in polar coordinates.

range_theta
: If provided, overrides auto-scaling on the angular axis in polar coordinates.

log_r
: If TRUE, the radial axis is log-scaled in polar coordinates.

render_mode
: One of "auto", "svg" or "webgl", default "auto" Controls the browser API used to draw marks. "svg" is appropriate for figures of less than 1000 data points, and will allow for fully-vectorized output. "webgl" is likely necessary for acceptable performance above 1000 points but rasterizes part of the output. "auto" uses heuristics to choose the mode.

title
: The figure title.

subtitle
: The figure subtitle.

template
: A template name (see template_names()) or a template list. Defaults to default_template().

width
: The figure width in pixels.

height
: The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_wind() |> px_scatter_polar(r = ~frequency, theta = ~direction, color = ~strength)
```

---

px_scatter_ternary *Ternary scatter plot*

Description

In a ternary scatter plot, each row of data_frame is represented by a symbol mark in ternary coordinates.

Usage

```
px_scatter_ternary(
  data_frame = NULL,
  a = NULL,
  b = NULL,
  c = NULL,
  color = NULL,
  symbol = NULL,
  size = NULL,
  text = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
```

```
  color_continuous_midpoint = NULL,
  symbol_sequence = NULL,
  symbol_map = NULL,
  opacity = NULL,
  size_max = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
a	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the a axis in ternary coordinates.
b	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the b axis in ternary coordinates.
c	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the c axis in ternary coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
symbol	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign symbols to marks.
size	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign mark sizes.
text	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.
hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and

values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data
: Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

animation_frame
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
: A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

color_continuous_scale
: Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color
: If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
: If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

symbol_sequence
: Strings should define valid plotly.js symbols. When symbol is set, values in that column are assigned symbols by cycling through symbol_sequence in the order described in category_orders, unless the value of symbol is a key in symbol_map.

symbol_map	Named character vector or list mapping values of symbol to marker symbols, overriding symbol_sequence. px_identity() uses the values of symbol directly.
opacity	Value between 0 and 1. Sets the opacity for markers.
size_max	Set the maximum mark size when using size.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_strip(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_election() |>
  px_scatter_ternary(a = ~Joly, b = ~Coderre, c = ~Bergeron, color = ~winner)
```

---

px_strip *Strip plot*

Description

In a strip plot each row of data_frame is represented as a jittered mark within categories.

Usage

```
px_strip(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  color = NULL,
  facet_row = NULL,
  facet_col = NULL,
```

```
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  orientation = NULL,
  stripmode = NULL,
  log_x = FALSE,
  log_y = FALSE,
  range_x = NULL,
  range_y = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
facet_row	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.
facet_col	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values

from this column or vector are used to assign marks to facetted subplots in the horizontal direction.

`facet_col_wrap` Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if `facet_row` or a `marginal` is set.

`facet_row_spacing`
Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

`facet_col_spacing`
Spacing between facet columns, in paper units Default is 0.02.

`hover_name` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector appear in bold in the hover tooltip.

`hover_data` Extra data to show in the hover tooltip: column names (a character vector or a formula such as `~c(a, b)`), or a named list with column names as names and values `TRUE` (default formatting), `FALSE` (hide the column, e.g. one shown by default), a d3-format string such as `":.3f"`, a vector of data, or a list of a format string or `TRUE` and a vector of data.

`custom_data` Column names (a character vector or a formula such as `~c(a, b)`), or a list of column names, formulas and vectors. Values from these columns are added to the traces' `customdata` (e.g. for use in Shiny events) without appearing in the hover tooltip.

`animation_frame`
A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to assign marks to animation frames.

`animation_group`
A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to provide object-constancy across animation frames: rows with matching `animation_groups` will be treated as if they describe the same object in each frame.

`category_orders`
Named list mapping column names to the order of their values in axes, legends and facets, e.g. `list(day = c("Thur", "Fri", "Sat", "Sun"))`. By default values appear in the order first encountered, or in level order for factors.

`labels` Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. `c(gdpPercap = "GDP per capita")`.

`color_discrete_sequence`
Character vector of colors, e.g. `px_colors$qualitative$Set1`. Values of `color` are assigned colors in order (cycling), unless set in `color_discrete_map`. Defaults to the template's `colorway`.

`color_discrete_map`
Named character vector or list mapping values of `color` to colors, overriding `color_discrete_sequence`. If the values of `color` are colors, `px_identity()` (or `"identity"`) uses them directly.

orientation	(default "v" if x and y are provided and both continuous or both categorical, otherwise "v"("h") if x(y) is categorical and y(x) is continuous, otherwise "v"("h") if only x(y) is provided)
stripmode	One of "group" or "overlay" In "overlay" mode, strips are on drawn top of one another. In "group" mode, strips are placed beside each other.
log_x	If TRUE, the x-axis is log-scaled in cartesian coordinates.
log_y	If TRUE, the y-axis is log-scaled in cartesian coordinates.
range_x	If provided, overrides auto-scaling on the x-axis in cartesian coordinates.
range_y	If provided, overrides auto-scaling on the y-axis in cartesian coordinates.
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_sunburst(), px_timeline(), px_treemap(), px_violin()

Examples

```
px_data_tips() |> px_strip(x = ~total_bill, y = ~day, color = ~sex)
```

---

px_sunburst *Sunburst chart*

Description

A sunburst plot represents hierarchial data as sectors laid out over several levels of concentric rings.

Usage

```
px_sunburst(
  data_frame = NULL,
  names = NULL,
  values = NULL,
  parents = NULL,
  path = NULL,
  ids = NULL,
  color = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  labels = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL,
  branchvalues = NULL,
  maxdepth = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
names	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used as labels for sectors.
values	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to set values associated to sectors.
parents	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used as parents in sunburst and treemap charts.
path	Column names in data_frame (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. The columns define the hierarchy of sectors, from root to leaves. Cannot be combined with ids or parents.
ids	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values

from this column or vector are used to set ids of sectors

`color` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to assign color to marks.

`color_continuous_scale`
Character vector of colors, e.g. `px_colors$sequential$Viridis`, or the name of a scale in px_colors. Used when `color` is numeric. Defaults to the template's sequential colorscale.

`range_color` If provided, overrides auto-scaling on the continuous color scale.

`color_continuous_midpoint`
If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using `plotly.express.colors.diverging` color scales as the inputs to `color_continuous_scale`.

`color_discrete_sequence`
Character vector of colors, e.g. `px_colors$qualitative$Set1`. Values of `color` are assigned colors in order (cycling), unless set in `color_discrete_map`. Defaults to the template's `colorway`.

`color_discrete_map`
Named character vector or list mapping values of `color` to colors, overriding `color_discrete_sequence`. If the values of `color` are colors, `px_identity()` (or `"identity"`) uses them directly.

`hover_name` A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector appear in bold in the hover tooltip.

`hover_data` Extra data to show in the hover tooltip: column names (a character vector or a formula such as `~c(a, b)`), or a named list with column names as names and values `TRUE` (default formatting), `FALSE` (hide the column, e.g. one shown by default), a d3-format string such as `":.3f"`, a vector of data, or a list of a format string or `TRUE` and a vector of data.

`custom_data` Column names (a character vector or a formula such as `~c(a, b)`), or a list of column names, formulas and vectors. Values from these columns are added to the traces' `customdata` (e.g. for use in Shiny events) without appearing in the hover tooltip.

`labels` Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. `c(gdpPercap = "GDP per capita")`.

`title` The figure title.

`subtitle` The figure subtitle.

`template` A template name (see `template_names()`) or a template list. Defaults to `default_template()`.

`width` The figure width in pixels.

`height` The figure height in pixels.

`branchvalues` "total" or "remainder" Determines how the items in `values` are summed. Whenset to "total", items in `values` are taken to be valueof all its descendants. When set to "remainder", itemsin `values` corresponding to the root and the branches:sectors are taken to be the extra part not part of thesum of the values at their leaves.

maxdepth Positive integer Sets the number of rendered sectors from any given `level`. Set `maxdepth` to -1 to render all thelevels in the hierarchy.

Value

A `go_figure()`.

See Also

Other plotly express: `px_area()`, `px_bar()`, `px_bar_polar()`, `px_box()`, `px_choropleth()`, `px_choropleth_map()`, `px_density_contour()`, `px_density_heatmap()`, `px_density_map()`, `px_ecdf()`, `px_funnel()`, `px_funnel_area()`, `px_histogram()`, `px_icicle()`, `px_imshow()`, `px_line()`, `px_line_3d()`, `px_line_geo()`, `px_line_map()`, `px_line_polar()`, `px_line_ternary()`, `px_parallel_categories()`, `px_parallel_coordinates()`, `px_pie()`, `px_scatter()`, `px_scatter_3d()`, `px_scatter_geo()`, `px_scatter_map()`, `px_scatter_matrix()`, `px_scatter_polar()`, `px_scatter_ternary()`, `px_strip()`, `px_timeline()`, `px_treemap()`, `px_violin()`

Examples

```
px_data_tips() |> px_sunburst(path = ~c(day, time, sex), values = ~total_bill)
```

---

px_timeline *Timeline (Gantt chart)*

Description

In a timeline plot, each row of `data_frame` is represented as a rectangular mark on an x axis of type `date`, spanning from `x_start` to `x_end`.

Usage

```
px_timeline(
  data_frame = NULL,
  x_start = NULL,
  x_end = NULL,
  y = NULL,
  color = NULL,
  pattern_shape = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  text = NULL,
  animation_frame = NULL,
```

```
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  pattern_shape_sequence = NULL,
  pattern_shape_map = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  opacity = NULL,
  range_x = NULL,
  range_y = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
x_start	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). (required) Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
x_end	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). (required) Values from this column or vector are used to position marks along the x axis in cartesian coordinates.
y	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
pattern_shape	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign pattern shapes to marks.
facet_row	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.

facet_col A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.

facet_col_wrap Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.

facet_row_spacing
Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

facet_col_spacing
Spacing between facet columns, in paper units Default is 0.02.

hover_name A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

text A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in the figure as text labels.

animation_frame
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

animation_group
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to provide object-constancy across animation frames: rows with matching animation_groups will be treated as if they describe the same object in each frame.

category_orders
Named list mapping column names to the order of their values in axes, legends and facets, e.g. list(day = c("Thur", "Fri", "Sat", "Sun")). By default values appear in the order first encountered, or in level order for factors.

labels Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").

color_discrete_sequence
: Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.

color_discrete_map
: Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.

pattern_shape_sequence
: Strings should define valid plotly.js patterns-shapes. When pattern_shape is set, values in that column are assigned patterns-shapes by cycling through pattern_shape_sequence in the order described in category_orders, unless the value of pattern_shape is a key in pattern_shape_map.

pattern_shape_map
: Named character vector or list mapping values of pattern_shape to pattern shapes, overriding pattern_shape_sequence. px_identity() uses the values directly.

color_continuous_scale
: Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.

range_color
: If provided, overrides auto-scaling on the continuous color scale.

color_continuous_midpoint
: If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.

opacity
: Value between 0 and 1. Sets the opacity for markers.

range_x
: If provided, overrides auto-scaling on the x-axis in cartesian coordinates.

range_y
: If provided, overrides auto-scaling on the y-axis in cartesian coordinates.

title
: The figure title.

subtitle
: The figure subtitle.

template
: A template name (see template_names()) or a template list. Defaults to default_template().

width
: The figure width in pixels.

height
: The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(),

px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_treemap(), px_violin()

Examples

```
data.frame(task = c("A", "B", "C"),
           start = c("2009-01-01", "2009-03-05", "2009-02-20"),
           finish = c("2009-02-28", "2009-04-15", "2009-05-30")) |>
  px_timeline(x_start = ~start, x_end = ~finish, y = ~task)
```

---

px_treemap *Treemap*

Description

A treemap plot represents hierarchial data as nested rectangular sectors.

Usage

```
px_treemap(
  data_frame = NULL,
  names = NULL,
  values = NULL,
  parents = NULL,
  ids = NULL,
  path = NULL,
  color = NULL,
  color_continuous_scale = NULL,
  range_color = NULL,
  color_continuous_midpoint = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  labels = NULL,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL,
  branchvalues = NULL,
  maxdepth = NULL
)
```

Arguments

data_frame	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if NULL, a data frame is built from the vectors given in the other arguments.
names	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used as labels for sectors.
values	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to set values associated to sectors.
parents	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used as parents in sunburst and treemap charts.
ids	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to set ids of sectors
path	Column names in data_frame (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. The columns define the hierarchy of sectors, from root to leaves. Cannot be combined with ids or parents.
color	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.
color_continuous_scale	Character vector of colors, e.g. px_colors$sequential$Viridis, or the name of a scale in px_colors. Used when color is numeric. Defaults to the template's sequential colorscale.
range_color	If provided, overrides auto-scaling on the continuous color scale.
color_continuous_midpoint	If set, computes the bounds of the continuous color scale to have the desired midpoint. Setting this value is recommended when using plotly.express.colors.diverging color scales as the inputs to color_continuous_scale.
color_discrete_sequence	Character vector of colors, e.g. px_colors$qualitative$Set1. Values of color are assigned colors in order (cycling), unless set in color_discrete_map. Defaults to the template's colorway.
color_discrete_map	Named character vector or list mapping values of color to colors, overriding color_discrete_sequence. If the values of color are colors, px_identity() (or "identity") uses them directly.
hover_name	A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data	Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.
custom_data	Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.
labels	Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. c(gdpPercap = "GDP per capita").
title	The figure title.
subtitle	The figure subtitle.
template	A template name (see template_names()) or a template list. Defaults to default_template().
width	The figure width in pixels.
height	The figure height in pixels.
branchvalues	"total" or "remainder" Determines how the items in values are summed. Whenset to "total", items in values are taken to be valueof all its descendants. When set to "remainder", itemsin values corresponding to the root and the branches:sectors are taken to be the extra part not part of thesum of the values at their leaves.
maxdepth	Positive integer Sets the number of rendered sectors from any given level. Set maxdepth to -1 to render all thelevels in the hierarchy.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_violin()

Examples

```
px_data_gapminder(year = 2007) |>
  px_treemap(path = list(px_constant("world"), ~continent, ~country),
             values = ~pop, color = ~lifeExp)
```

px_violin *Violin plot*

Description

In a violin plot, rows of `data_frame` are grouped together into a curved mark to visualize their distribution.

Usage

```
px_violin(
  data_frame = NULL,
  x = NULL,
  y = NULL,
  color = NULL,
  facet_row = NULL,
  facet_col = NULL,
  facet_col_wrap = 0,
  facet_row_spacing = NULL,
  facet_col_spacing = NULL,
  hover_name = NULL,
  hover_data = NULL,
  custom_data = NULL,
  animation_frame = NULL,
  animation_group = NULL,
  category_orders = NULL,
  labels = NULL,
  color_discrete_sequence = NULL,
  color_discrete_map = NULL,
  orientation = NULL,
  violinmode = NULL,
  log_x = FALSE,
  log_y = FALSE,
  range_x = NULL,
  range_y = NULL,
  points = NULL,
  box = FALSE,
  title = NULL,
  subtitle = NULL,
  template = NULL,
  width = NULL,
  height = NULL
)
```

Arguments

`data_frame`	A data.frame, data.table, list or matrix. Needed for column names to be used in the other arguments; if `NULL`, a data frame is built from the vectors given in the

other arguments.

x A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the x axis in cartesian coordinates.

y A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to position marks along the y axis in cartesian coordinates.

color A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign color to marks.

facet_row A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the vertical direction.

facet_col A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to facetted subplots in the horizontal direction.

facet_col_wrap Maximum number of facet columns. Wraps the column variable at this width, so that the column facets span multiple rows. Ignored if 0, and forced to 0 if facet_row or a marginal is set.

facet_row_spacing
Spacing between facet rows, in paper units. Default is 0.03 or 0.07 when facet_col_wrap is used.

facet_col_spacing
Spacing between facet columns, in paper units Default is 0.02.

hover_name A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector appear in bold in the hover tooltip.

hover_data Extra data to show in the hover tooltip: column names (a character vector or a formula such as ~c(a, b)), or a named list with column names as names and values TRUE (default formatting), FALSE (hide the column, e.g. one shown by default), a d3-format string such as ":.3f", a vector of data, or a list of a format string or TRUE and a vector of data.

custom_data Column names (a character vector or a formula such as ~c(a, b)), or a list of column names, formulas and vectors. Values from these columns are added to the traces' customdata (e.g. for use in Shiny events) without appearing in the hover tooltip.

animation_frame
A column name in data_frame, a vector of values, a one-sided formula evaluated in data_frame (e.g. ~ log(pop)), px_constant() or px_range(). Values from this column or vector are used to assign marks to animation frames.

`animation_group`
: A column name in `data_frame`, a vector of values, a one-sided formula evaluated in `data_frame` (e.g. `~ log(pop)`), `px_constant()` or `px_range()`. Values from this column or vector are used to provide object-constancy across animation frames: rows with matching `animation_groups` will be treated as if they describe the same object in each frame.

`category_orders`
: Named list mapping column names to the order of their values in axes, legends and facets, e.g. `list(day = c("Thur", "Fri", "Sat", "Sun"))`. By default values appear in the order first encountered, or in level order for factors.

`labels`
: Named character vector or list mapping column names to the labels to use for axis titles, legend entries and hover text, e.g. `c(gdpPercap = "GDP per capita")`.

`color_discrete_sequence`
: Character vector of colors, e.g. `px_colors$qualitative$Set1`. Values of `color` are assigned colors in order (cycling), unless set in `color_discrete_map`. Defaults to the template's `colorway`.

`color_discrete_map`
: Named character vector or list mapping values of `color` to colors, overriding `color_discrete_sequence`. If the values of `color` are colors, `px_identity()` (or `"identity"`) uses them directly.

`orientation`
: (default `"v"` if `x` and `y` are provided and both continuous or both categorical, otherwise `"v"`(`"h"`) if `x`(`y`) is categorical and `y`(`x`) is continuous, otherwise `"v"`(`"h"`) if only `x`(`y`) is provided)

`violinmode`
: One of `"group"` or `"overlay"` In `"overlay"` mode, violins are on drawn top of one another. In `"group"` mode, violins are placed beside each other.

`log_x`
: If `TRUE`, the x-axis is log-scaled in cartesian coordinates.

`log_y`
: If `TRUE`, the y-axis is log-scaled in cartesian coordinates.

`range_x`
: If provided, overrides auto-scaling on the x-axis in cartesian coordinates.

`range_y`
: If provided, overrides auto-scaling on the y-axis in cartesian coordinates.

`points`
: One of `"outliers"`, `"suspectedoutliers"`, `"all"`, or `FALSE`. If `"outliers"`, only the sample points lying outside the whiskers are shown. If `"suspectedoutliers"`, all outlier points are shown and those less than 4*Q1-3*Q3 or greater than 4*Q3-3*Q1 are highlighted with the marker's `"outliercolor"`. If `"outliers"`, only the sample points lying outside the whiskers are shown. If `"all"`, all sample points are shown. If `FALSE`, no sample points are shown and the whiskers extend to the full range of the sample.

`box`
: If `TRUE`, boxes are drawn inside the violins.

`title`
: The figure title.

`subtitle`
: The figure subtitle.

`template`
: A template name (see `template_names()`) or a template list. Defaults to `default_template()`.

`width`
: The figure width in pixels.

`height`
: The figure height in pixels.

Value

A go_figure().

See Also

Other plotly express: px_area(), px_bar(), px_bar_polar(), px_box(), px_choropleth(), px_choropleth_map(), px_density_contour(), px_density_heatmap(), px_density_map(), px_ecdf(), px_funnel(), px_funnel_area(), px_histogram(), px_icicle(), px_imshow(), px_line(), px_line_3d(), px_line_geo(), px_line_map(), px_line_polar(), px_line_ternary(), px_parallel_categories(), px_parallel_coordinates(), px_pie(), px_scatter(), px_scatter_3d(), px_scatter_geo(), px_scatter_map(), px_scatter_matrix(), px_scatter_polar(), px_scatter_ternary(), px_strip(), px_sunburst(), px_timeline(), px_treemap()

Examples

```
px_data_tips() |> px_violin(y = ~total_bill, x = ~day, color = ~smoker, box = TRUE)
```

---

schema *The plotly.js plot schema*

Description

The schema of the bundled plotly.js: every trace type and layout object with its attributes.

Usage

```
schema(jsonedit = interactive(), ...)
```

Arguments

jsonedit	Show it in an interactive viewer (needs the listviewer package)?
...	Unused.

Value

The schema as a list (invisibly): traces, layout, frames, animation, config and defs.

Examples

```
s <- schema(jsonedit = FALSE)
names(s$traces)
```

select_traces *Select, update or modify traces*

Description

`select_traces()` returns the traces matching `selector` (and, for figures created with `make_subplots()`, the grid cell given by `row`, `col` and `secondary_y`). `update_traces()` updates their properties like `update_layout()` does. `for_each_trace()` replaces each one with the result of `fn`.

Usage

```
select_traces(fig, selector = NULL, row = NULL, col = NULL, secondary_y = NULL)

update_traces(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  overwrite = FALSE
)

for_each_trace(
  fig,
  fn,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL
)
```

Arguments

`fig`	A `go_figure` object.
`selector`	Which traces to use: `NULL` (all), a trace type such as `"scatter"`, indices into the matching traces, a named list of property values that must all match (names may use magic underscores, e.g. `list(marker_color = "red")`), or a function that takes a trace and returns `TRUE` or `FALSE`.
`row`, `col`	Only traces in this subplot grid row and/or column.
`secondary_y`	Only traces on secondary (`TRUE`) or primary (`FALSE`) y axes.
`patch`	A named list of property updates.
`...`	Further named updates.
`overwrite`	If `TRUE`, replace nested objects instead of merging.

fn	A function that takes a trace and returns the modified trace. If it returns anything other than a list, the trace is left unchanged.

Value

`select_traces()` returns a list of traces; the others return the modified figure.

Examples

```
fig <- go_figure() |>
  add_scatter(y = 1:3, name = "a") |>
  add_scatter(y = 3:1, name = "b", mode = "markers")
select_traces(fig, list(name = "b"))
fig <- update_traces(fig, marker_size = 12, selector = list(mode = "markers"))
fig <- for_each_trace(fig, function(tr) {
  tr$name <- toupper(tr$name)
  tr
})
```

---

set_crosstalk_keys *Link traces to a crosstalk group*

Description

Gives traces of a figure the keys of a crosstalk::SharedData, so that selecting their points selects the same rows in other views of the crosstalk group (and the reverse). This happens by itself for `px_*()` functions given a SharedData as `data_frame`, and for `add_<trace>()` traces whose formula properties are evaluated in a SharedData (piped into `go_figure()` or given as `data`); use this for other traces. See `highlight()` for how selections are made and shown.

Usage

```
set_crosstalk_keys(fig, key, group = NULL, traces = NULL, simple = FALSE)
```

Arguments

fig	A `go_figure()`.
key	A crosstalk::SharedData, whose row keys give the keys of the points, in the same order as the trace's data; or the keys (a vector with one per point), or a list with the keys of each trace in `traces`.
group	The crosstalk group. Default: the SharedData's group.
traces	Indices of the traces to link (default: all).
simple	Whether each trace as a whole stands for all its keys (e.g. a line or a fitted curve, which can't be split into points); selecting the trace selects them all, and it is highlighted when all are selected.

Value

`fig`, linked.

Examples

```
sd <- crosstalk::SharedData$new(px_data_iris())
iris <- sd$origData()
go_figure() |>
  add_scatter(x = iris$sepal_width, y = iris$sepal_length, mode = "markers") |>
  set_crosstalk_keys(sd) |>
  highlight(on = "plotly_selected")
```

TeX *Wrap text for LaTeX rendering*

Description

Wraps a string in `$...$` so plotly.js renders it with MathJax (requires `config(p, mathjax = "local")`, or `"cdn"`; `save_image()` needs nothing).

Usage

```
TeX(x)
```

Arguments

`x`	A character string of LaTeX.

Value

The wrapped string.

Examples

```
go_figure() |>
  update_layout(title = TeX("\\alpha + \\beta x")) |>
  config(mathjax = "local")
```

update_annotations *Select, update or modify annotations*

Description

`select_annotations()` returns the figure's annotations matching `selector` (and grid cell); `update_annotations()` updates their properties like `update_layout()` does; `for_each_annotation()` replaces each one with the result of `fn`.

Usage

```
select_annotations(
  fig,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL
)

update_annotations(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  overwrite = FALSE
)

for_each_annotation(
  fig,
  fn,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL
)
```

Arguments

`fig`	A `go_figure` object.
`selector`	Which objects to use: `NULL` (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns `TRUE` or `FALSE`.
`row, col`	Only objects on this subplot grid row and/or column's axes.

secondary_y	Only objects on secondary (TRUE) or primary (FALSE) y axes.
patch	A named list of property updates (see go_layout_annotation()).
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_annotations() returns a list of go_layout_annotation() objects; the others return the modified figure.

See Also

add_annotation()

---

update_coloraxes *Select, update or modify coloraxis objects*

Description

Work with all of a figure's layout coloraxis objects (coloraxis, coloraxis2, coloraxis3, ...). select_coloraxes() returns them as a named list, update_coloraxes() updates their properties like update_layout() does, and for_each_coloraxis() replaces each one with the result of fn. The base object (e.g. coloraxis) is always included, even if it has not been set yet.

Usage

```
select_coloraxes(fig, selector = NULL, row = NULL, col = NULL)

update_coloraxes(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  overwrite = FALSE
)

for_each_coloraxis(fig, fn, selector = NULL, row = NULL, col = NULL)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects in this subplot grid row and/or column (figures created with make_subplots() only).
patch	A named list or go_layout_coloraxis() object of property updates.
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_coloraxes() returns a named list of go_layout_coloraxis() objects; the others return the modified figure.

---

update_geos *Select, update or modify geo objects*

Description

Work with all of a figure's layout geo objects (geo, geo2, geo3, ...). select_geos() returns them as a named list, update_geos() updates their properties like update_layout() does, and for_each_geo() replaces each one with the result of fn. The base object (e.g. geo) is always included, even if it has not been set yet.

Usage

```
select_geos(fig, selector = NULL, row = NULL, col = NULL)

update_geos(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  overwrite = FALSE
)

for_each_geo(fig, fn, selector = NULL, row = NULL, col = NULL)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects in this subplot grid row and/or column (figures created with make_subplots() only).
patch	A named list or go_layout_geo() object of property updates.
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_geos() returns a named list of go_layout_geo() objects; the others return the modified figure.

---

update_layout *Update a figure's layout*

Description

Updates layout properties, merging nested objects (e.g. xaxis) with any existing values unless overwrite = TRUE. Properties can be given in patch, in ..., or both; ... is applied last. Names may use "magic underscores": xaxis_title_text = "x" is the same as xaxis = list(title = list(text = "x")).

Usage

```
update_layout(fig, patch = NULL, ..., overwrite = FALSE)
```

Arguments

fig	A go_figure object.
patch	A named list or go_layout() object of updates.
...	Further named updates.
overwrite	If TRUE, replace nested objects instead of merging.

Value

The modified figure.

Examples

```
go_figure() |>
  update_layout(title = "Title", xaxis = list(range = c(0, 10))) |>
  update_layout(xaxis_title = "x", yaxis = list(type = "log"))
```

`update_layout_images` *Select, update or modify images*

Description

`select_layout_images()` returns the figure's images matching `selector` (and grid cell); `update_layout_images()` updates their properties like `update_layout()` does; `for_each_layout_image()` replaces each one with the result of `fn`.

Usage

```
select_layout_images(
  fig,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL
)

update_layout_images(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  overwrite = FALSE
)

for_each_layout_image(
  fig,
  fn,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL
)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects on this subplot grid row and/or column's axes.
secondary_y	Only objects on secondary (TRUE) or primary (FALSE) y axes.
patch	A named list of property updates (see go_layout_image()).
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_layout_images() returns a list of go_layout_image() objects; the others return the modified figure.

See Also

add_layout_image()

---

update_legends *Select, update or modify legend objects*

Description

Work with all of a figure's layout legend objects (legend, legend2, legend3, ...). select_legends() returns them as a named list, update_legends() updates their properties like update_layout() does, and for_each_legend() replaces each one with the result of fn. The base object (e.g. legend) is always included, even if it has not been set yet.

Usage

```
select_legends(fig, selector = NULL, row = NULL, col = NULL)

update_legends(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  overwrite = FALSE
)

for_each_legend(fig, fn, selector = NULL, row = NULL, col = NULL)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects in this subplot grid row and/or column (figures created with make_subplots() only).
patch	A named list or go_layout_legend() object of property updates.
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_legends() returns a named list of go_layout_legend() objects; the others return the modified figure.

---

update_maps *Select, update or modify map objects*

Description

Work with all of a figure's layout map objects (map, map2, map3, ...). select_maps() returns them as a named list, update_maps() updates their properties like update_layout() does, and for_each_map() replaces each one with the result of fn. The base object (e.g. map) is always included, even if it has not been set yet.

Usage

```
select_maps(fig, selector = NULL, row = NULL, col = NULL)

update_maps(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  overwrite = FALSE
)

for_each_map(fig, fn, selector = NULL, row = NULL, col = NULL)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects in this subplot grid row and/or column (figures created with make_subplots() only).
patch	A named list or go_layout_map() object of property updates.
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_maps() returns a named list of go_layout_map() objects; the others return the modified figure.

---

update_polars *Select, update or modify polar objects*

Description

Work with all of a figure's layout polar objects (polar, polar2, polar3, ...). select_polars() returns them as a named list, update_polars() updates their properties like update_layout() does, and for_each_polar() replaces each one with the result of fn. The base object (e.g. polar) is always included, even if it has not been set yet.

Usage

```
select_polars(fig, selector = NULL, row = NULL, col = NULL)

update_polars(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  overwrite = FALSE
)

for_each_polar(fig, fn, selector = NULL, row = NULL, col = NULL)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects in this subplot grid row and/or column (figures created with make_subplots() only).
patch	A named list or go_layout_polar() object of property updates.
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_polars() returns a named list of go_layout_polar() objects; the others return the modified figure.

---

update_scenes *Select, update or modify scene objects*

Description

Work with all of a figure's layout scene objects (scene, scene2, scene3, ...). select_scenes() returns them as a named list, update_scenes() updates their properties like update_layout() does, and for_each_scene() replaces each one with the result of fn. The base object (e.g. scene) is always included, even if it has not been set yet.

Usage

```
select_scenes(fig, selector = NULL, row = NULL, col = NULL)

update_scenes(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  overwrite = FALSE
)

for_each_scene(fig, fn, selector = NULL, row = NULL, col = NULL)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects in this subplot grid row and/or column (figures created with make_subplots() only).
patch	A named list or go_layout_scene() object of property updates.
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_scenes() returns a named list of go_layout_scene() objects; the others return the modified figure.

---

update_selections *Select, update or modify selections*

Description

select_selections() returns the figure's selections matching selector (and grid cell); update_selections() updates their properties like update_layout() does; for_each_selection() replaces each one with the result of fn.

Usage

```
select_selections(
  fig,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL
)

update_selections(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
```

```
  overwrite = FALSE
)

for_each_selection(
  fig,
  fn,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL
)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects on this subplot grid row and/or column's axes.
secondary_y	Only objects on secondary (TRUE) or primary (FALSE) y axes.
patch	A named list of property updates (see go_layout_selection()).
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_selections() returns a list of go_layout_selection() objects; the others return the modified figure.

See Also

add_selection()

---

update_shapes *Select, update or modify shapes*

Description

select_shapes() returns the figure's shapes matching selector (and grid cell); update_shapes() updates their properties like update_layout() does; for_each_shape() replaces each one with the result of fn.

Usage

```
select_shapes(fig, selector = NULL, row = NULL, col = NULL, secondary_y = NULL)

update_shapes(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  overwrite = FALSE
)

for_each_shape(
  fig,
  fn,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL
)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects on this subplot grid row and/or column's axes.
secondary_y	Only objects on secondary (TRUE) or primary (FALSE) y axes.
patch	A named list of property updates (see go_layout_shape()).
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_shapes() returns a list of go_layout_shape() objects; the others return the modified figure.

See Also

add_shape()

update_smiths *Select, update or modify smith objects*

Description

Work with all of a figure's layout smith objects (smith, smith2, smith3, ...). select_smiths() returns them as a named list, update_smiths() updates their properties like update_layout() does, and for_each_smith() replaces each one with the result of fn. The base object (e.g. smith) is always included, even if it has not been set yet.

Usage

```
select_smiths(fig, selector = NULL, row = NULL, col = NULL)

update_smiths(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  overwrite = FALSE
)

for_each_smith(fig, fn, selector = NULL, row = NULL, col = NULL)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects in this subplot grid row and/or column (figures created with make_subplots() only).
patch	A named list or go_layout_smith() object of property updates.
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_smiths() returns a named list of go_layout_smith() objects; the others return the modified figure.

update_ternaries *Select, update or modify ternary objects*

Description

Work with all of a figure's layout ternary objects (`ternary`, `ternary2`, `ternary3`, ...). `select_ternaries()` returns them as a named list, `update_ternaries()` updates their properties like `update_layout()` does, and `for_each_ternary()` replaces each one with the result of `fn`. The base object (e.g. `ternary`) is always included, even if it has not been set yet.

Usage

```
select_ternaries(fig, selector = NULL, row = NULL, col = NULL)

update_ternaries(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  overwrite = FALSE
)

for_each_ternary(fig, fn, selector = NULL, row = NULL, col = NULL)
```

Arguments

`fig`	A `go_figure` object.
`selector`	Which objects to use: `NULL` (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns `TRUE` or `FALSE`.
`row, col`	Only objects in this subplot grid row and/or column (figures created with `make_subplots()` only).
`patch`	A named list or `go_layout_ternary()` object of property updates.
`...`	Further named updates (names may use magic underscores).
`overwrite`	If `TRUE`, replace nested objects instead of merging.
`fn`	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

`select_ternaries()` returns a named list of `go_layout_ternary()` objects; the others return the modified figure.

update_xaxes *Select, update or modify xaxis objects*

Description

Work with all of a figure's layout xaxis objects (xaxis, xaxis2, xaxis3, ...). select_xaxes() returns them as a named list, update_xaxes() updates their properties like update_layout() does, and for_each_xaxis() replaces each one with the result of fn. The base object (e.g. xaxis) is always included, even if it has not been set yet.

Usage

```
select_xaxes(fig, selector = NULL, row = NULL, col = NULL)

update_xaxes(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  overwrite = FALSE
)

for_each_xaxis(fig, fn, selector = NULL, row = NULL, col = NULL)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects in this subplot grid row and/or column (figures created with make_subplots() only).
patch	A named list or go_layout_xaxis() object of property updates.
...	Further named updates (names may use magic underscores).
overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_xaxes() returns a named list of go_layout_xaxis() objects; the others return the modified figure.

update_yaxes *Select, update or modify yaxis objects*

Description

Work with all of a figure's layout yaxis objects (yaxis, yaxis2, yaxis3, ...). select_yaxes() returns them as a named list, update_yaxes() updates their properties like update_layout() does, and for_each_yaxis() replaces each one with the result of fn. The base object (e.g. yaxis) is always included, even if it has not been set yet.

Usage

```
select_yaxes(fig, selector = NULL, row = NULL, col = NULL, secondary_y = NULL)

update_yaxes(
  fig,
  patch = NULL,
  ...,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL,
  overwrite = FALSE
)

for_each_yaxis(
  fig,
  fn,
  selector = NULL,
  row = NULL,
  col = NULL,
  secondary_y = NULL
)
```

Arguments

fig	A go_figure object.
selector	Which objects to use: NULL (all), indices into the matching objects, a named list of property values that must all match, or a function that takes an object and returns TRUE or FALSE.
row, col	Only objects in this subplot grid row and/or column (figures created with make_subplots() only).
secondary_y	Only secondary (TRUE) or primary (FALSE) y axes.
patch	A named list or go_layout_yaxis() object of property updates.
...	Further named updates (names may use magic underscores).

overwrite	If TRUE, replace nested objects instead of merging.
fn	A function that takes an object and returns the modified object. If it returns anything other than a list, the object is left unchanged.

Value

select_yaxes() returns a named list of go_layout_yaxis() objects; the others return the modified figure.

---

write_html *Write a figure to an HTML file*

Description

Saves a figure as a standalone web page, like htmlwidgets::saveWidget() (and plotly.py's fig.write_html()), but faster for figures with a lot of data: the page is laid out by htmlwidgets as usual, while the figure's data is written to the file directly, rather than passed through htmltools as one long string, which is scanned and copied several times. For a heatmap of 4 million values this takes a fraction of the time saveWidget() does.

Usage

```
write_html(
  fig,
  file,
  selfcontained = TRUE,
  libdir = NULL,
  background = "white",
  title = NULL
)
```

Arguments

fig	A figure (e.g. from go_figure() or px_scatter()) or a plotly.core widget (e.g. from plotly_widget(), or a figure from plotly's plot_ly()).
file	Path of the HTML file to write.
selfcontained	Whether to embed the files the page needs in it, so that the file works on its own. Otherwise they are written to libdir.
libdir	Directory for those files when selfcontained = FALSE, relative to the directory of file (default: <name of file>_files).
background	Background color of the page.
title	Title of the page (default: the widget's name).

Details

The page is the same as `saveWidget()` writes, except that with `selfcontained = TRUE` the files the page needs (plotly.js, htmlwidgets and so on) are embedded as base64 data URIs by `write_html()` itself, without needing pandoc.

Value

`file`, invisibly.

See Also

`htmlwidgets::saveWidget()`, `save_image()` for static images.

Examples

```
fig <- go_figure() |> add_heatmap(z = as_float32(volcano))
f <- tempfile(fileext = ".html")
write_html(fig, f)
```

Index

∗ **datasets**
px_colors, 1133
px_defaults, 1136
∗ **plotly express**
px_area, 1111
px_bar, 1115
px_bar_polar, 1119
px_box, 1123
px_choropleth, 1126
px_choropleth_map, 1130
px_density_contour, 1137
px_density_heatmap, 1141
px_density_map, 1145
px_ecdf, 1147
px_funnel, 1152
px_funnel_area, 1155
px_histogram, 1158
px_icicle, 1162
px_imshow, 1164
px_line, 1167
px_line_3d, 1171
px_line_geo, 1175
px_line_map, 1180
px_line_polar, 1183
px_line_ternary, 1186
px_parallel_categories, 1190
px_parallel_coordinates, 1191
px_pie, 1193
px_scatter, 1196
px_scatter_3d, 1201
px_scatter_geo, 1205
px_scatter_map, 1209
px_scatter_matrix, 1213
px_scatter_polar, 1215
px_scatter_ternary, 1219
px_strip, 1222
px_sunburst, 1225
px_timeline, 1228
px_treemap, 1232
px_violin, 1235
[[<-.plotly_obj *(plotly_obj-modify)*, 1104
$<-.plotly_obj *(plotly_obj-modify)*, 1104

add_annotation, 5, *1243*
add_annotation(), *1100*
add_bar *(go_bar)*, 746
add_barpolar *(go_barpolar)*, 772
add_box *(go_box)*, 792
add_candlestick *(go_candlestick)*, 809
add_carpet *(go_carpet)*, 819
add_choropleth *(go_choropleth)*, 834
add_choroplethmap *(go_choroplethmap)*, 851
add_cone *(go_cone)*, 867
add_contour *(go_contour)*, 885
add_contourcarpet *(go_contourcarpet)*, 906
add_densitymap *(go_densitymap)*, 922
add_funnel, 14
add_funnelarea, 37
add_heatmap, 49
add_histogram, 68
add_histogram2d, 96
add_histogram2dcontour, 116
add_icicle, 138
add_image, 160
add_indicator, 168
add_isosurface, 183
add_layout_image, 204, *1247*
add_mesh3d, 207
add_ohlc, 226
add_parcats, 237
add_parcoords, 252
add_pie, 267
add_quiver, 281
add_sankey, 301
add_scatter, 314
add_scatter(), *665*

add_scatter3d, 345
add_scattercarpet, 372
add_scattergeo, 397
add_scattergl, 420
add_scattermap, 447
add_scatterpolar, 466
add_scatterpolargl, 490
add_scattersmith, 514
add_scatterternary, 538
add_selection, 562, *1252*
add_shape, 565, *1253*
add_splom, 574
add_streamtube, 595
add_sunburst, 613
add_surface, 633
add_table, 654
add_trace, *36*, *49*, *68*, *96*, *116*, *138*, *160*, *168*, *182*, *204*, *226*, *237*, *252*, *267*, *280*, *301*, *314*, *345*, *372*, *396*, *420*, *446*, *466*, *490*, *513*, *537*, *562*, *595*, *613*, *633*, *654*, *663*, *687*, *702*, *724*, *740*, *772*, *792*, *808*, *819*, *834*, *850*, *867*, *885*, *906*, *921*, *937*
add_trace *(*add_traces*)*, 664
add_trace(), *937*, *1100*
add_traces, 664
add_treemap, 665
add_violin, 687
add_volume, 703
add_waterfall, 724
as_float32, 740
as_float32(), *1106*
as_widget *(*plotly_widget*)*, 1105
as_widget(), *741*, *1104*
attrs_selected *(*highlight*)*, 1097

chrome_exporter, 741
config, 742
config(), *1105*
crosstalk::filter_select(), *1097*
crosstalk::SharedData, *937*, *1097*, *1240*

default_template *(*get_template*)*, 745
default_template(), *1106*, *1114*, *1119*, *1122*, *1126*, *1130*, *1133*, *1140*, *1144*, *1147*, *1151*, *1154*, *1156*, *1161*, *1164*, *1171*, *1175*, *1179*, *1182*, *1186*, *1189*, *1191*, *1192*, *1195*, *1200*, *1204*, *1209*, *1212*, *1215*, *1218*, *1222*, *1225*, *1227*, *1231*, *1234*, *1237*

event_data, 743
event_data(), *740*, *744*, *745*, *1107*
event_register, 744
event_source *(*event_register*)*, 744
event_source(), *743*
event_unregister *(*event_register*)*, 744

for_each_annotation *(*update_annotations*)*, 1242
for_each_coloraxis *(*update_coloraxes*)*, 1243
for_each_geo *(*update_geos*)*, 1244
for_each_layout_image *(*update_layout_images*)*, 1246
for_each_legend *(*update_legends*)*, 1247
for_each_map *(*update_maps*)*, 1248
for_each_polar *(*update_polars*)*, 1249
for_each_scene *(*update_scenes*)*, 1250
for_each_selection *(*update_selections*)*, 1251
for_each_shape *(*update_shapes*)*, 1252
for_each_smith *(*update_smiths*)*, 1254
for_each_ternary *(*update_ternaries*)*, 1255
for_each_trace *(*select_traces*)*, 1239
for_each_xaxis *(*update_xaxes*)*, 1256
for_each_yaxis *(*update_yaxes*)*, 1257

get_subplot *(*print_grid*)*, 1110
get_subplot(), *1102*
get_template, 745
go_bar, 746
go_bar_error_x, *756*
go_bar_error_x *(*go_bar*)*, 746
go_bar_error_y, *756*
go_bar_error_y *(*go_bar*)*, 746
go_bar_hoverlabel, *756*
go_bar_hoverlabel *(*go_bar*)*, 746
go_bar_hoverlabel_font, *763*
go_bar_hoverlabel_font *(*go_bar*)*, 746
go_bar_insidetextfont, *757*
go_bar_insidetextfont *(*go_bar*)*, 746
go_bar_legendgrouptitle, *757*
go_bar_legendgrouptitle *(*go_bar*)*, 746
go_bar_legendgrouptitle_font, *763*

go_bar_legendgrouptitle_font (go_bar), 746
go_bar_marker, *758*
go_bar_marker (go_bar), 746
go_bar_marker_colorbar, *766*
go_bar_marker_colorbar (go_bar), 746
go_bar_marker_colorbar_tickfont, *768*
go_bar_marker_colorbar_tickfont (go_bar), 746
go_bar_marker_colorbar_tickformatstop, *768*
go_bar_marker_colorbar_tickformatstop (go_bar), 746
go_bar_marker_colorbar_title, *769*
go_bar_marker_colorbar_title (go_bar), 746
go_bar_marker_colorbar_title_font, *763*
go_bar_marker_colorbar_title_font (go_bar), 746
go_bar_marker_line, *767*
go_bar_marker_line (go_bar), 746
go_bar_marker_pattern, *767*
go_bar_marker_pattern (go_bar), 746
go_bar_outsidetextfont, *758*
go_bar_outsidetextfont (go_bar), 746
go_bar_selected, *758*
go_bar_selected (go_bar), 746
go_bar_selected_marker, *758*
go_bar_selected_marker (go_bar), 746
go_bar_selected_textfont, *759*
go_bar_selected_textfont (go_bar), 746
go_bar_textfont, *759*
go_bar_textfont (go_bar), 746
go_bar_unselected, *760*
go_bar_unselected (go_bar), 746
go_bar_unselected_marker, *758*
go_bar_unselected_marker (go_bar), 746
go_bar_unselected_textfont, *759*
go_bar_unselected_textfont (go_bar), 746
go_barpolar, 772
go_barpolar_hoverlabel, *779*
go_barpolar_hoverlabel (go_barpolar), 772
go_barpolar_hoverlabel_font, *783*
go_barpolar_hoverlabel_font (go_barpolar), 772
go_barpolar_legendgrouptitle, *780*
go_barpolar_legendgrouptitle (go_barpolar), 772
go_barpolar_legendgrouptitle_font, *783*
go_barpolar_legendgrouptitle_font (go_barpolar), 772
go_barpolar_marker, *780*
go_barpolar_marker (go_barpolar), 772
go_barpolar_marker_colorbar, *786*
go_barpolar_marker_colorbar (go_barpolar), 772
go_barpolar_marker_colorbar_tickfont, *788*
go_barpolar_marker_colorbar_tickfont (go_barpolar), 772
go_barpolar_marker_colorbar_tickformatstop, *788*
go_barpolar_marker_colorbar_tickformatstop (go_barpolar), 772
go_barpolar_marker_colorbar_title, *789*
go_barpolar_marker_colorbar_title (go_barpolar), 772
go_barpolar_marker_colorbar_title_font, *783*
go_barpolar_marker_colorbar_title_font (go_barpolar), 772
go_barpolar_marker_line, *786*
go_barpolar_marker_line (go_barpolar), 772
go_barpolar_marker_pattern, *786*
go_barpolar_marker_pattern (go_barpolar), 772
go_barpolar_selected, *781*
go_barpolar_selected (go_barpolar), 772
go_barpolar_selected_marker, *780*
go_barpolar_selected_marker (go_barpolar), 772
go_barpolar_selected_textfont, *791*
go_barpolar_selected_textfont (go_barpolar), 772
go_barpolar_unselected, *782*
go_barpolar_unselected (go_barpolar), 772
go_barpolar_unselected_marker, *781*
go_barpolar_unselected_marker (go_barpolar), 772
go_barpolar_unselected_textfont, *791*
go_barpolar_unselected_textfont (go_barpolar), 772
go_box, 792

go_box_hoverlabel, *798*
go_box_hoverlabel (go_box), 792
go_box_hoverlabel_font, *804*
go_box_hoverlabel_font (go_box), 792
go_box_legendgrouptitle, *799*
go_box_legendgrouptitle (go_box), 792
go_box_legendgrouptitle_font, *804*
go_box_legendgrouptitle_font (go_box), 792
go_box_line, *800*
go_box_line (go_box), 792
go_box_marker, *800*
go_box_marker (go_box), 792
go_box_marker_line, *800*
go_box_marker_line (go_box), 792
go_box_selected, *801*
go_box_selected (go_box), 792
go_box_selected_marker, *800*
go_box_selected_marker (go_box), 792
go_box_unselected, *802*
go_box_unselected (go_box), 792
go_box_unselected_marker, *800*
go_box_unselected_marker (go_box), 792
go_candlestick, 809
go_candlestick_decreasing, *813*
go_candlestick_decreasing (go_candlestick), 809
go_candlestick_decreasing_line, *814*
go_candlestick_decreasing_line (go_candlestick), 809
go_candlestick_hoverlabel, *813*
go_candlestick_hoverlabel (go_candlestick), 809
go_candlestick_hoverlabel_font, *817*
go_candlestick_hoverlabel_font (go_candlestick), 809
go_candlestick_increasing, *814*
go_candlestick_increasing (go_candlestick), 809
go_candlestick_increasing_line, *814*
go_candlestick_increasing_line (go_candlestick), 809
go_candlestick_legendgrouptitle, *814*
go_candlestick_legendgrouptitle (go_candlestick), 809
go_candlestick_legendgrouptitle_font, *817*
go_candlestick_legendgrouptitle_font (go_candlestick), 809
go_candlestick_line, *814*
go_candlestick_line (go_candlestick), 809
go_carpet, 819
go_carpet_aaxis, *827*
go_carpet_aaxis (go_carpet), 819
go_carpet_aaxis_tickfont, *831*
go_carpet_aaxis_tickfont (go_carpet), 819
go_carpet_aaxis_tickformatstop, *832*
go_carpet_aaxis_tickformatstop (go_carpet), 819
go_carpet_aaxis_title, *832*
go_carpet_aaxis_title (go_carpet), 819
go_carpet_aaxis_title_font, *827*
go_carpet_aaxis_title_font (go_carpet), 819
go_carpet_baxis, *827*
go_carpet_baxis (go_carpet), 819
go_carpet_baxis_tickfont, *831*
go_carpet_baxis_tickfont (go_carpet), 819
go_carpet_baxis_tickformatstop, *832*
go_carpet_baxis_tickformatstop (go_carpet), 819
go_carpet_baxis_title, *832*
go_carpet_baxis_title (go_carpet), 819
go_carpet_baxis_title_font, *827*
go_carpet_baxis_title_font (go_carpet), 819
go_carpet_font, *827*
go_carpet_font (go_carpet), 819
go_carpet_legendgrouptitle, *828*
go_carpet_legendgrouptitle (go_carpet), 819
go_carpet_legendgrouptitle_font, *827*
go_carpet_legendgrouptitle_font (go_carpet), 819
go_choropleth, 834
go_choropleth_colorbar, *840*
go_choropleth_colorbar (go_choropleth), 834
go_choropleth_colorbar_tickfont, *845*
go_choropleth_colorbar_tickfont (go_choropleth), 834
go_choropleth_colorbar_tickformatstop, *845*

go_choropleth_colorbar_tickformatstop (go_choropleth), 834
go_choropleth_colorbar_title, *846*
go_choropleth_colorbar_title (go_choropleth), 834
go_choropleth_colorbar_title_font, *849*
go_choropleth_colorbar_title_font (go_choropleth), 834
go_choropleth_hoverlabel, *841*
go_choropleth_hoverlabel (go_choropleth), 834
go_choropleth_hoverlabel_font, *849*
go_choropleth_hoverlabel_font (go_choropleth), 834
go_choropleth_legendgrouptitle, *842*
go_choropleth_legendgrouptitle (go_choropleth), 834
go_choropleth_legendgrouptitle_font, *849*
go_choropleth_legendgrouptitle_font (go_choropleth), 834
go_choropleth_marker, *842*
go_choropleth_marker (go_choropleth), 834
go_choropleth_marker_line, *849*
go_choropleth_marker_line (go_choropleth), 834
go_choropleth_selected, *843*
go_choropleth_selected (go_choropleth), 834
go_choropleth_selected_marker, *842*
go_choropleth_selected_marker (go_choropleth), 834
go_choropleth_unselected, *843*
go_choropleth_unselected (go_choropleth), 834
go_choropleth_unselected_marker, *842*
go_choropleth_unselected_marker (go_choropleth), 834
go_choroplethmap, 851
go_choroplethmap_colorbar, *857*
go_choroplethmap_colorbar (go_choroplethmap), 851
go_choroplethmap_colorbar_tickfont, *862*
go_choroplethmap_colorbar_tickfont (go_choroplethmap), 851
go_choroplethmap_colorbar_tickformatstop, *862*
go_choroplethmap_colorbar_tickformatstop (go_choroplethmap), 851
go_choroplethmap_colorbar_title, *863*
go_choroplethmap_colorbar_title (go_choroplethmap), 851
go_choroplethmap_colorbar_title_font, *865*
go_choroplethmap_colorbar_title_font (go_choroplethmap), 851
go_choroplethmap_hoverlabel, *858*
go_choroplethmap_hoverlabel (go_choroplethmap), 851
go_choroplethmap_hoverlabel_font, *865*
go_choroplethmap_hoverlabel_font (go_choroplethmap), 851
go_choroplethmap_legendgrouptitle, *858*
go_choroplethmap_legendgrouptitle (go_choroplethmap), 851
go_choroplethmap_legendgrouptitle_font, *866*
go_choroplethmap_legendgrouptitle_font (go_choroplethmap), 851
go_choroplethmap_marker, *859*
go_choroplethmap_marker (go_choroplethmap), 851
go_choroplethmap_marker_line, *866*
go_choroplethmap_marker_line (go_choroplethmap), 851
go_choroplethmap_selected, *859*
go_choroplethmap_selected (go_choroplethmap), 851
go_choroplethmap_selected_marker, *859*
go_choroplethmap_selected_marker (go_choroplethmap), 851
go_choroplethmap_unselected, *860*
go_choroplethmap_unselected (go_choroplethmap), 851
go_choroplethmap_unselected_marker, *859*
go_choroplethmap_unselected_marker (go_choroplethmap), 851
go_cone, 867
go_cone_colorbar, *874*
go_cone_colorbar (go_cone), 867
go_cone_colorbar_tickfont, *880*
go_cone_colorbar_tickfont (go_cone), 867
go_cone_colorbar_tickformatstop, *880*

go_cone_colorbar_tickformatstop (go_cone), 867
go_cone_colorbar_title, *881*
go_cone_colorbar_title (go_cone), 867
go_cone_colorbar_title_font, *883*
go_cone_colorbar_title_font (go_cone), 867
go_cone_hoverlabel, *875*
go_cone_hoverlabel (go_cone), 867
go_cone_hoverlabel_font, *883*
go_cone_hoverlabel_font (go_cone), 867
go_cone_legendgrouptitle, *875*
go_cone_legendgrouptitle (go_cone), 867
go_cone_legendgrouptitle_font, *883*
go_cone_legendgrouptitle_font (go_cone), 867
go_cone_lighting, *876*
go_cone_lighting (go_cone), 867
go_cone_lightposition, *876*
go_cone_lightposition (go_cone), 867
go_contour, 885
go_contour_colorbar, *893*
go_contour_colorbar (go_contour), 885
go_contour_colorbar_tickfont, *900*
go_contour_colorbar_tickfont (go_contour), 885
go_contour_colorbar_tickformatstop, *900*
go_contour_colorbar_tickformatstop (go_contour), 885
go_contour_colorbar_title, *901*
go_contour_colorbar_title (go_contour), 885
go_contour_colorbar_title_font, *904*
go_contour_colorbar_title_font (go_contour), 885
go_contour_contours, *893*
go_contour_contours (go_contour), 885
go_contour_contours_labelfont, *904*
go_contour_contours_labelfont (go_contour), 885
go_contour_hoverlabel, *894*
go_contour_hoverlabel (go_contour), 885
go_contour_hoverlabel_font, *904*
go_contour_hoverlabel_font (go_contour), 885
go_contour_legendgrouptitle, *895*
go_contour_legendgrouptitle (go_contour), 885
go_contour_legendgrouptitle_font, *904*
go_contour_legendgrouptitle_font (go_contour), 885
go_contour_line, *895*
go_contour_line (go_contour), 885
go_contour_textfont, *896*
go_contour_textfont (go_contour), 885
go_contourcarpet, 906
go_contourcarpet_colorbar, *913*
go_contourcarpet_colorbar (go_contourcarpet), 906
go_contourcarpet_colorbar_tickfont, *917*
go_contourcarpet_colorbar_tickfont (go_contourcarpet), 906
go_contourcarpet_colorbar_tickformatstop, *917*
go_contourcarpet_colorbar_tickformatstop (go_contourcarpet), 906
go_contourcarpet_colorbar_title, *918*
go_contourcarpet_colorbar_title (go_contourcarpet), 906
go_contourcarpet_colorbar_title_font, *920*
go_contourcarpet_colorbar_title_font (go_contourcarpet), 906
go_contourcarpet_contours, *913*
go_contourcarpet_contours (go_contourcarpet), 906
go_contourcarpet_contours_labelfont, *920*
go_contourcarpet_contours_labelfont (go_contourcarpet), 906
go_contourcarpet_legendgrouptitle, *913*
go_contourcarpet_legendgrouptitle (go_contourcarpet), 906
go_contourcarpet_legendgrouptitle_font, *920*
go_contourcarpet_legendgrouptitle_font (go_contourcarpet), 906
go_contourcarpet_line, *914*
go_contourcarpet_line (go_contourcarpet), 906
go_densitymap, 922
go_densitymap_colorbar, *927*
go_densitymap_colorbar (go_densitymap), 922

go_densitymap_colorbar_tickfont, *932*
go_densitymap_colorbar_tickfont (go_densitymap), 922
go_densitymap_colorbar_tickformatstop, *932*
go_densitymap_colorbar_tickformatstop (go_densitymap), 922
go_densitymap_colorbar_title, *933*
go_densitymap_colorbar_title (go_densitymap), 922
go_densitymap_colorbar_title_font, *935*
go_densitymap_colorbar_title_font (go_densitymap), 922
go_densitymap_hoverlabel, *928*
go_densitymap_hoverlabel (go_densitymap), 922
go_densitymap_hoverlabel_font, *935*
go_densitymap_hoverlabel_font (go_densitymap), 922
go_densitymap_legendgrouptitle, *929*
go_densitymap_legendgrouptitle (go_densitymap), 922
go_densitymap_legendgrouptitle_font, *935*
go_densitymap_legendgrouptitle_font (go_densitymap), 922
go_figure, *22*, *36*, *42*, *49*, *56*, *68*, *78*, *96*, *103*, *116*, *124*, *138*, *147*, *160*, *163*, *168*, *175*, *182*, *191*, *204*, *214*, *226*, *230*, *237*, *242*, *252*, *258*, *267*, *273*, *280*, *288*, *301*, *307*, *314*, *325*, *345*, *356*, *372*, *381*, *396*, *405*, *420*, *430*, *446*, *454*, *466*, *474*, *490*, *498*, *513*, *522*, *537*, *546*, *562*, *581*, *595*, *602*, *613*, *621*, *633*, *641*, *654*, *658*, *663*, *673*, *687*, *693*, *702*, *711*, *724*, *731*, *740*, *770*, *772*, *791, 792*, *807, 808*, *818, 819*, *833, 834*, *849, 850*, *866, 867*, *884, 885*, *905, 906*, *920, 921*, *936*, 937, *937*, *947*
go_figure(), *741*, *743*, *745*, *1097, 1098*, *1102–1105*, *1107*, *1114*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1154*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1179*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*, *1240*, *1258*
go_funnel (add_funnel), 14
go_funnel_connector, *22*
go_funnel_connector (add_funnel), 14
go_funnel_connector_line, *28*
go_funnel_connector_line (add_funnel), 14
go_funnel_hoverlabel, *22*
go_funnel_hoverlabel (add_funnel), 14
go_funnel_hoverlabel_font, *29*
go_funnel_hoverlabel_font (add_funnel), 14
go_funnel_insidetextfont, *23*
go_funnel_insidetextfont (add_funnel), 14
go_funnel_legendgrouptitle, *23*
go_funnel_legendgrouptitle (add_funnel), 14
go_funnel_legendgrouptitle_font, *29*
go_funnel_legendgrouptitle_font (add_funnel), 14
go_funnel_marker, *24*
go_funnel_marker (add_funnel), 14
go_funnel_marker_colorbar, *31*
go_funnel_marker_colorbar (add_funnel), 14
go_funnel_marker_colorbar_tickfont, *33*
go_funnel_marker_colorbar_tickfont (add_funnel), 14
go_funnel_marker_colorbar_tickformatstop, *34*
go_funnel_marker_colorbar_tickformatstop (add_funnel), 14
go_funnel_marker_colorbar_title, *35*
go_funnel_marker_colorbar_title (add_funnel), 14
go_funnel_marker_colorbar_title_font, *29*
go_funnel_marker_colorbar_title_font (add_funnel), 14
go_funnel_marker_line, *28*
go_funnel_marker_line (add_funnel), 14
go_funnel_outsidetextfont, *24*
go_funnel_outsidetextfont (add_funnel), 14
go_funnel_textfont, *25*
go_funnel_textfont (add_funnel), 14
go_funnelarea (add_funnelarea), 37
go_funnelarea_domain, *42*

go_funnelarea_domain (add_funnelarea), 37
go_funnelarea_hoverlabel, *42*
go_funnelarea_hoverlabel (add_funnelarea), 37
go_funnelarea_hoverlabel_font, *46*
go_funnelarea_hoverlabel_font (add_funnelarea), 37
go_funnelarea_insidetextfont, *43*
go_funnelarea_insidetextfont (add_funnelarea), 37
go_funnelarea_legendgrouptitle, *43*
go_funnelarea_legendgrouptitle (add_funnelarea), 37
go_funnelarea_legendgrouptitle_font, *46*
go_funnelarea_legendgrouptitle_font (add_funnelarea), 37
go_funnelarea_marker, *44*
go_funnelarea_marker (add_funnelarea), 37
go_funnelarea_marker_line, *48*
go_funnelarea_marker_line (add_funnelarea), 37
go_funnelarea_marker_pattern, *48*
go_funnelarea_marker_pattern (add_funnelarea), 37
go_funnelarea_textfont, *44*
go_funnelarea_textfont (add_funnelarea), 37
go_funnelarea_title, *45*
go_funnelarea_title (add_funnelarea), 37
go_funnelarea_title_font, *46*
go_funnelarea_title_font (add_funnelarea), 37
go_heatmap (add_heatmap), 49
go_heatmap_colorbar, *56*
go_heatmap_colorbar (add_heatmap), 49
go_heatmap_colorbar_tickfont, *63*
go_heatmap_colorbar_tickfont (add_heatmap), 49
go_heatmap_colorbar_tickformatstop, *63*
go_heatmap_colorbar_tickformatstop (add_heatmap), 49
go_heatmap_colorbar_title, *64*
go_heatmap_colorbar_title (add_heatmap), 49
go_heatmap_colorbar_title_font, *66*
go_heatmap_colorbar_title_font (add_heatmap), 49
go_heatmap_hoverlabel, *57*
go_heatmap_hoverlabel (add_heatmap), 49
go_heatmap_hoverlabel_font, *66*
go_heatmap_hoverlabel_font (add_heatmap), 49
go_heatmap_legendgrouptitle, *58*
go_heatmap_legendgrouptitle (add_heatmap), 49
go_heatmap_legendgrouptitle_font, *67*
go_heatmap_legendgrouptitle_font (add_heatmap), 49
go_heatmap_textfont, *58*
go_heatmap_textfont (add_heatmap), 49
go_histogram (add_histogram), 68
go_histogram2d (add_histogram2d), 96
go_histogram2d_colorbar, *104*
go_histogram2d_colorbar (add_histogram2d), 96
go_histogram2d_colorbar_tickfont, *110*
go_histogram2d_colorbar_tickfont (add_histogram2d), 96
go_histogram2d_colorbar_tickformatstop, *111*
go_histogram2d_colorbar_tickformatstop (add_histogram2d), 96
go_histogram2d_colorbar_title, *112*
go_histogram2d_colorbar_title (add_histogram2d), 96
go_histogram2d_colorbar_title_font, *114*
go_histogram2d_colorbar_title_font (add_histogram2d), 96
go_histogram2d_hoverlabel, *105*
go_histogram2d_hoverlabel (add_histogram2d), 96
go_histogram2d_hoverlabel_font, *114*
go_histogram2d_hoverlabel_font (add_histogram2d), 96
go_histogram2d_legendgrouptitle, *105*
go_histogram2d_legendgrouptitle (add_histogram2d), 96
go_histogram2d_legendgrouptitle_font, *114*
go_histogram2d_legendgrouptitle_font (add_histogram2d), 96
go_histogram2d_marker, *106*

go_histogram2d_marker
(add_histogram2d), 96
go_histogram2d_textfont, *106*
go_histogram2d_textfont
(add_histogram2d), 96
go_histogram2d_xbins, *107*
go_histogram2d_xbins (add_histogram2d),
96
go_histogram2d_ybins, *108*
go_histogram2d_ybins (add_histogram2d),
96
go_histogram2dcontour
(add_histogram2dcontour), 116
go_histogram2dcontour_colorbar, *125*
go_histogram2dcontour_colorbar
(add_histogram2dcontour), 116
go_histogram2dcontour_colorbar_tickfont,
131
go_histogram2dcontour_colorbar_tickfont
(add_histogram2dcontour), 116
go_histogram2dcontour_colorbar_tickformatstop,
132
go_histogram2dcontour_colorbar_tickformatstop
(add_histogram2dcontour), 116
go_histogram2dcontour_colorbar_title,
132
go_histogram2dcontour_colorbar_title
(add_histogram2dcontour), 116
go_histogram2dcontour_colorbar_title_font,
135
go_histogram2dcontour_colorbar_title_font
(add_histogram2dcontour), 116
go_histogram2dcontour_contours, *125*
go_histogram2dcontour_contours
(add_histogram2dcontour), 116
go_histogram2dcontour_contours_labelfont,
136
go_histogram2dcontour_contours_labelfont
(add_histogram2dcontour), 116
go_histogram2dcontour_hoverlabel, *125*
go_histogram2dcontour_hoverlabel
(add_histogram2dcontour), 116
go_histogram2dcontour_hoverlabel_font,
135
go_histogram2dcontour_hoverlabel_font
(add_histogram2dcontour), 116
go_histogram2dcontour_legendgrouptitle,
126
go_histogram2dcontour_legendgrouptitle
(add_histogram2dcontour), 116
go_histogram2dcontour_legendgrouptitle_font,
135
go_histogram2dcontour_legendgrouptitle_font
(add_histogram2dcontour), 116
go_histogram2dcontour_line, *126*
go_histogram2dcontour_line
(add_histogram2dcontour), 116
go_histogram2dcontour_marker, *126*
go_histogram2dcontour_marker
(add_histogram2dcontour), 116
go_histogram2dcontour_textfont, *127*
go_histogram2dcontour_textfont
(add_histogram2dcontour), 116
go_histogram2dcontour_xbins, *128*
go_histogram2dcontour_xbins
(add_histogram2dcontour), 116
go_histogram2dcontour_ybins, *129*
go_histogram2dcontour_ybins
(add_histogram2dcontour), 116
go_histogram_cumulative, *79*
go_histogram_cumulative
(add_histogram), 68
go_histogram_error_x, *79*
go_histogram_error_x (add_histogram), 68
go_histogram_error_y, *79*
go_histogram_error_y (add_histogram), 68
go_histogram_hoverlabel, *79*
go_histogram_hoverlabel
(add_histogram), 68
go_histogram_hoverlabel_font, *87*
go_histogram_hoverlabel_font
(add_histogram), 68
go_histogram_insidetextfont, *80*
go_histogram_insidetextfont
(add_histogram), 68
go_histogram_legendgrouptitle, *80*
go_histogram_legendgrouptitle
(add_histogram), 68
go_histogram_legendgrouptitle_font, *87*
go_histogram_legendgrouptitle_font
(add_histogram), 68
go_histogram_marker, *81*
go_histogram_marker (add_histogram), 68
go_histogram_marker_colorbar, *90*
go_histogram_marker_colorbar
(add_histogram), 68

go_histogram_marker_colorbar_tickfont, *92*
go_histogram_marker_colorbar_tickfont *(*add_histogram*)*, 68
go_histogram_marker_colorbar_tickformatstop, *92*
go_histogram_marker_colorbar_tickformatstop *(*add_histogram*)*, 68
go_histogram_marker_colorbar_title, *93*
go_histogram_marker_colorbar_title *(*add_histogram*)*, 68
go_histogram_marker_colorbar_title_font, *87*
go_histogram_marker_colorbar_title_font *(*add_histogram*)*, 68
go_histogram_marker_line, *90*
go_histogram_marker_line *(*add_histogram*)*, 68
go_histogram_marker_pattern, *90*
go_histogram_marker_pattern *(*add_histogram*)*, 68
go_histogram_outsidetextfont, *82*
go_histogram_outsidetextfont *(*add_histogram*)*, 68
go_histogram_selected, *82*
go_histogram_selected *(*add_histogram*)*, 68
go_histogram_selected_marker, *81*
go_histogram_selected_marker *(*add_histogram*)*, 68
go_histogram_selected_textfont, *82*
go_histogram_selected_textfont *(*add_histogram*)*, 68
go_histogram_textfont, *82*
go_histogram_textfont *(*add_histogram*)*, 68
go_histogram_unselected, *83*
go_histogram_unselected *(*add_histogram*)*, 68
go_histogram_unselected_marker, *81*
go_histogram_unselected_marker *(*add_histogram*)*, 68
go_histogram_unselected_textfont, *82*
go_histogram_unselected_textfont *(*add_histogram*)*, 68
go_histogram_xbins, *83*
go_histogram_xbins *(*add_histogram*)*, 68
go_histogram_ybins, *84*
go_histogram_ybins *(*add_histogram*)*, 68
go_icicle *(*add_icicle*)*, 138
go_icicle_domain, *147*
go_icicle_domain *(*add_icicle*)*, 138
go_icicle_hoverlabel, *147*
go_icicle_hoverlabel *(*add_icicle*)*, 138
go_icicle_hoverlabel_font, *152*
go_icicle_hoverlabel_font *(*add_icicle*)*, 138
go_icicle_insidetextfont, *148*
go_icicle_insidetextfont *(*add_icicle*)*, 138
go_icicle_leaf, *148*
go_icicle_leaf *(*add_icicle*)*, 138
go_icicle_legendgrouptitle, *148*
go_icicle_legendgrouptitle *(*add_icicle*)*, 138
go_icicle_legendgrouptitle_font, *152*
go_icicle_legendgrouptitle_font *(*add_icicle*)*, 138
go_icicle_marker, *148*
go_icicle_marker *(*add_icicle*)*, 138
go_icicle_marker_colorbar, *154*
go_icicle_marker_colorbar *(*add_icicle*)*, 138
go_icicle_marker_colorbar_tickfont, *156*
go_icicle_marker_colorbar_tickfont *(*add_icicle*)*, 138
go_icicle_marker_colorbar_tickformatstop, *156*
go_icicle_marker_colorbar_tickformatstop *(*add_icicle*)*, 138
go_icicle_marker_colorbar_title, *157*
go_icicle_marker_colorbar_title *(*add_icicle*)*, 138
go_icicle_marker_colorbar_title_font, *152*
go_icicle_marker_colorbar_title_font *(*add_icicle*)*, 138
go_icicle_marker_line, *154*
go_icicle_marker_line *(*add_icicle*)*, 138
go_icicle_marker_pattern, *154*
go_icicle_marker_pattern *(*add_icicle*)*, 138
go_icicle_outsidetextfont, *149*
go_icicle_outsidetextfont *(*add_icicle*)*, 138

go_icicle_pathbar, *149*
go_icicle_pathbar *(*add_icicle*)*, 138
go_icicle_pathbar_textfont, *149*
go_icicle_pathbar_textfont *(*add_icicle*)*, 138
go_icicle_root, *149*
go_icicle_root *(*add_icicle*)*, 138
go_icicle_textfont, *149*
go_icicle_textfont *(*add_icicle*)*, 138
go_icicle_tiling, *150*
go_icicle_tiling *(*add_icicle*)*, 138
go_image *(*add_image*)*, 160
go_image_hoverlabel, *163*
go_image_hoverlabel *(*add_image*)*, 160
go_image_hoverlabel_font, *166*
go_image_hoverlabel_font *(*add_image*)*, 160
go_image_legendgrouptitle, *164*
go_image_legendgrouptitle *(*add_image*)*, 160
go_image_legendgrouptitle_font, *166*
go_image_legendgrouptitle_font *(*add_image*)*, 160
go_indicator *(*add_indicator*)*, 168
go_indicator_delta, *175*
go_indicator_delta *(*add_indicator*)*, 168
go_indicator_delta_decreasing, *177*
go_indicator_delta_decreasing *(*add_indicator*)*, 168
go_indicator_delta_font, *177*
go_indicator_delta_font *(*add_indicator*)*, 168
go_indicator_delta_increasing, *177*
go_indicator_delta_increasing *(*add_indicator*)*, 168
go_indicator_domain, *175*
go_indicator_domain *(*add_indicator*)*, 168
go_indicator_gauge, *175*
go_indicator_gauge *(*add_indicator*)*, 168
go_indicator_gauge_axis, *179*
go_indicator_gauge_axis *(*add_indicator*)*, 168
go_indicator_gauge_axis_tickfont, *180*
go_indicator_gauge_axis_tickfont *(*add_indicator*)*, 168
go_indicator_gauge_axis_tickformatstop, *180*
go_indicator_gauge_axis_tickformatstop *(*add_indicator*)*, 168
go_indicator_gauge_bar, *179*
go_indicator_gauge_bar *(*add_indicator*)*, 168
go_indicator_gauge_bar_line, *181*
go_indicator_gauge_bar_line *(*add_indicator*)*, 168
go_indicator_gauge_step, *179*
go_indicator_gauge_step *(*add_indicator*)*, 168
go_indicator_gauge_step_line, *181*
go_indicator_gauge_step_line *(*add_indicator*)*, 168
go_indicator_gauge_threshold, *179*
go_indicator_gauge_threshold *(*add_indicator*)*, 168
go_indicator_gauge_threshold_line, *181*
go_indicator_gauge_threshold_line *(*add_indicator*)*, 168
go_indicator_legendgrouptitle, *176*
go_indicator_legendgrouptitle *(*add_indicator*)*, 168
go_indicator_legendgrouptitle_font, *177*
go_indicator_legendgrouptitle_font *(*add_indicator*)*, 168
go_indicator_number, *176*
go_indicator_number *(*add_indicator*)*, 168
go_indicator_number_font, *177*
go_indicator_number_font *(*add_indicator*)*, 168
go_indicator_title, *176*
go_indicator_title *(*add_indicator*)*, 168
go_indicator_title_font, *177*
go_indicator_title_font *(*add_indicator*)*, 168
go_isosurface *(*add_isosurface*)*, 183
go_isosurface_caps, *191*
go_isosurface_caps *(*add_isosurface*)*, 183
go_isosurface_caps_x, *195*
go_isosurface_caps_x *(*add_isosurface*)*, 183
go_isosurface_caps_y, *195*
go_isosurface_caps_y *(*add_isosurface*)*, 183
go_isosurface_caps_z, *196*
go_isosurface_caps_z *(*add_isosurface*)*, 183

go_isosurface_colorbar, *192*
go_isosurface_colorbar (add_isosurface), 183
go_isosurface_colorbar_tickfont, *199*
go_isosurface_colorbar_tickfont (add_isosurface), 183
go_isosurface_colorbar_tickformatstop, *199*
go_isosurface_colorbar_tickformatstop (add_isosurface), 183
go_isosurface_colorbar_title, *200*
go_isosurface_colorbar_title (add_isosurface), 183
go_isosurface_colorbar_title_font, *202*
go_isosurface_colorbar_title_font (add_isosurface), 183
go_isosurface_contour, *192*
go_isosurface_contour (add_isosurface), 183
go_isosurface_hoverlabel, *192*
go_isosurface_hoverlabel (add_isosurface), 183
go_isosurface_hoverlabel_font, *202*
go_isosurface_hoverlabel_font (add_isosurface), 183
go_isosurface_legendgrouptitle, *193*
go_isosurface_legendgrouptitle (add_isosurface), 183
go_isosurface_legendgrouptitle_font, *202*
go_isosurface_legendgrouptitle_font (add_isosurface), 183
go_isosurface_lighting, *193*
go_isosurface_lighting (add_isosurface), 183
go_isosurface_lightposition, *193*
go_isosurface_lightposition (add_isosurface), 183
go_isosurface_slices, *194*
go_isosurface_slices (add_isosurface), 183
go_isosurface_slices_x, *195*
go_isosurface_slices_x (add_isosurface), 183
go_isosurface_slices_y, *195*
go_isosurface_slices_y (add_isosurface), 183
go_isosurface_slices_z, *196*
go_isosurface_slices_z (add_isosurface), 183
go_isosurface_spaceframe, *194*
go_isosurface_spaceframe (add_isosurface), 183
go_isosurface_surface, *194*
go_isosurface_surface (add_isosurface), 183
go_layout, *14*, *207*, *565*, *574*, 938, *948*, *949*, *957*, *958*, *960*, *966*, *968*, *971*, *977*, *983*, *985*, *986*, *988*, *993*, *1006*, *1028*, *1033*, *1039*, *1052*, *1056–1058*, *1063*, *1081*, *1097*
go_layout(), *937*, *1245*
go_layout_activeselection, *940*, 947
go_layout_activeshape, *940*, 948
go_layout_annotation, *940*, *1243*
go_layout_annotation (add_annotation), 5
go_layout_annotation_font, *10*
go_layout_annotation_font (add_annotation), 5
go_layout_annotation_font(), *1102*
go_layout_annotation_hoverlabel, *10*
go_layout_annotation_hoverlabel (add_annotation), 5
go_layout_annotation_hoverlabel_font, *10*
go_layout_annotation_hoverlabel_font (add_annotation), 5
go_layout_coloraxis, *941*, 949, *1244*
go_layout_coloraxis_colorbar, *952*
go_layout_coloraxis_colorbar (go_layout_coloraxis), 949
go_layout_coloraxis_colorbar_tickfont, *954*
go_layout_coloraxis_colorbar_tickfont (go_layout_coloraxis), 949
go_layout_coloraxis_colorbar_tickformatstop, *954*
go_layout_coloraxis_colorbar_tickformatstop (go_layout_coloraxis), 949
go_layout_coloraxis_colorbar_title, *955*
go_layout_coloraxis_colorbar_title (go_layout_coloraxis), 949
go_layout_coloraxis_colorbar_title_font, *957*
go_layout_coloraxis_colorbar_title_font

(go_layout_coloraxis), 949
go_layout_colorscale, *942*, 957
go_layout_font, *943*, 959
go_layout_geo, *943*, 960, *1245*
go_layout_geo_center, *963*
go_layout_geo_center (go_layout_geo), 960
go_layout_geo_domain, *963*
go_layout_geo_domain (go_layout_geo), 960
go_layout_geo_lataxis, *963*
go_layout_geo_lataxis (go_layout_geo), 960
go_layout_geo_lonaxis, *963*
go_layout_geo_lonaxis (go_layout_geo), 960
go_layout_geo_projection, *963*
go_layout_geo_projection (go_layout_geo), 960
go_layout_geo_projection_rotation, *965*
go_layout_geo_projection_rotation (go_layout_geo), 960
go_layout_grid, *943*, 966
go_layout_grid_domain, *967*
go_layout_grid_domain (go_layout_grid), 966
go_layout_hoverlabel, *943*, 969
go_layout_hoverlabel_font, *970*
go_layout_hoverlabel_font (go_layout_hoverlabel), 969
go_layout_hoverlabel_grouptitlefont, *970*
go_layout_hoverlabel_grouptitlefont (go_layout_hoverlabel), 969
go_layout_image, *944*, *1247*
go_layout_image (add_layout_image), 204
go_layout_legend, *944*, 971, *1248*
go_layout_legend_font, *974*
go_layout_legend_font (go_layout_legend), 971
go_layout_legend_grouptitlefont, *974*
go_layout_legend_grouptitlefont (go_layout_legend), 971
go_layout_legend_title, *975*
go_layout_legend_title (go_layout_legend), 971
go_layout_legend_title_font, *974*
go_layout_legend_title_font (go_layout_legend), 971
go_layout_map, *944*, 977, *1249*
go_layout_map_bounds, *980*
go_layout_map_bounds (go_layout_map), 977
go_layout_map_center, *980*
go_layout_map_center (go_layout_map), 977
go_layout_map_domain, *980*
go_layout_map_domain (go_layout_map), 977
go_layout_map_layer, *980*
go_layout_map_layer (go_layout_map), 977
go_layout_map_layer_circle, *981*
go_layout_map_layer_circle (go_layout_map), 977
go_layout_map_layer_fill, *981*
go_layout_map_layer_fill (go_layout_map), 977
go_layout_map_layer_line, *981*
go_layout_map_layer_line (go_layout_map), 977
go_layout_map_layer_symbol, *982*
go_layout_map_layer_symbol (go_layout_map), 977
go_layout_map_layer_symbol_textfont, *983*
go_layout_map_layer_symbol_textfont (go_layout_map), 977
go_layout_margin, *944*, 984
go_layout_modebar, *944*, 985
go_layout_newselection, *944*, 986
go_layout_newselection_line, *987*
go_layout_newselection_line (go_layout_newselection), 986
go_layout_newshape, *944*, 988
go_layout_newshape_label, *990*
go_layout_newshape_label (go_layout_newshape), 988
go_layout_newshape_label_font, *991*
go_layout_newshape_label_font (go_layout_newshape), 988
go_layout_newshape_legendgrouptitle, *990*
go_layout_newshape_legendgrouptitle (go_layout_newshape), 988
go_layout_newshape_legendgrouptitle_font, *991*

go_layout_newshape_legendgrouptitle_font
(go_layout_newshape), 988
go_layout_newshape_line, *990*
go_layout_newshape_line
(go_layout_newshape), 988
go_layout_polar, *945*, 993, *1250*
go_layout_polar_angularaxis, *998*
go_layout_polar_angularaxis
(go_layout_polar), 993
go_layout_polar_angularaxis_tickfont,
1002
go_layout_polar_angularaxis_tickfont
(go_layout_polar), 993
go_layout_polar_angularaxis_tickformatstop,
1002
go_layout_polar_angularaxis_tickformatstop
(go_layout_polar), 993
go_layout_polar_domain, *998*
go_layout_polar_domain
(go_layout_polar), 993
go_layout_polar_radialaxis, *998*
go_layout_polar_radialaxis
(go_layout_polar), 993
go_layout_polar_radialaxis_autorangeoptions,
1004
go_layout_polar_radialaxis_autorangeoptions
(go_layout_polar), 993
go_layout_polar_radialaxis_tickfont,
1002
go_layout_polar_radialaxis_tickfont
(go_layout_polar), 993
go_layout_polar_radialaxis_tickformatstop,
1002
go_layout_polar_radialaxis_tickformatstop
(go_layout_polar), 993
go_layout_polar_radialaxis_title, *1005*
go_layout_polar_radialaxis_title
(go_layout_polar), 993
go_layout_polar_radialaxis_title_font,
1005
go_layout_polar_radialaxis_title_font
(go_layout_polar), 993
go_layout_scene, *945*, 1006, *1251*
go_layout_scene_annotation, *1017*
go_layout_scene_annotation
(go_layout_scene), 1006
go_layout_scene_annotation_font, *1018*
go_layout_scene_annotation_font
(go_layout_scene), 1006
go_layout_scene_annotation_hoverlabel,
1019
go_layout_scene_annotation_hoverlabel
(go_layout_scene), 1006
go_layout_scene_annotation_hoverlabel_font,
1018
go_layout_scene_annotation_hoverlabel_font
(go_layout_scene), 1006
go_layout_scene_aspectratio, *1017*
go_layout_scene_aspectratio
(go_layout_scene), 1006
go_layout_scene_camera, *1017*
go_layout_scene_camera
(go_layout_scene), 1006
go_layout_scene_camera_center, *1021*
go_layout_scene_camera_center
(go_layout_scene), 1006
go_layout_scene_camera_eye, *1021*
go_layout_scene_camera_eye
(go_layout_scene), 1006
go_layout_scene_camera_projection,
1021
go_layout_scene_camera_projection
(go_layout_scene), 1006
go_layout_scene_camera_up, *1021*
go_layout_scene_camera_up
(go_layout_scene), 1006
go_layout_scene_domain, *1017*
go_layout_scene_domain
(go_layout_scene), 1006
go_layout_scene_xaxis, *1018*
go_layout_scene_xaxis
(go_layout_scene), 1006
go_layout_scene_xaxis_autorangeoptions,
1022
go_layout_scene_xaxis_autorangeoptions
(go_layout_scene), 1006
go_layout_scene_xaxis_tickfont, *1025*
go_layout_scene_xaxis_tickfont
(go_layout_scene), 1006
go_layout_scene_xaxis_tickformatstop,
1025
go_layout_scene_xaxis_tickformatstop
(go_layout_scene), 1006
go_layout_scene_xaxis_title, *1026*
go_layout_scene_xaxis_title
(go_layout_scene), 1006

go_layout_scene_xaxis_title_font, *1018*
go_layout_scene_xaxis_title_font (go_layout_scene), 1006
go_layout_scene_yaxis, *1018*
go_layout_scene_yaxis (go_layout_scene), 1006
go_layout_scene_yaxis_autorangeoptions, *1022*
go_layout_scene_yaxis_autorangeoptions (go_layout_scene), 1006
go_layout_scene_yaxis_tickfont, *1025*
go_layout_scene_yaxis_tickfont (go_layout_scene), 1006
go_layout_scene_yaxis_tickformatstop, *1025*
go_layout_scene_yaxis_tickformatstop (go_layout_scene), 1006
go_layout_scene_yaxis_title, *1026*
go_layout_scene_yaxis_title (go_layout_scene), 1006
go_layout_scene_yaxis_title_font, *1018*
go_layout_scene_yaxis_title_font (go_layout_scene), 1006
go_layout_scene_zaxis, *1018*
go_layout_scene_zaxis (go_layout_scene), 1006
go_layout_scene_zaxis_autorangeoptions, *1022*
go_layout_scene_zaxis_autorangeoptions (go_layout_scene), 1006
go_layout_scene_zaxis_tickfont, *1025*
go_layout_scene_zaxis_tickfont (go_layout_scene), 1006
go_layout_scene_zaxis_tickformatstop, *1025*
go_layout_scene_zaxis_tickformatstop (go_layout_scene), 1006
go_layout_scene_zaxis_title, *1026*
go_layout_scene_zaxis_title (go_layout_scene), 1006
go_layout_scene_zaxis_title_font, *1018*
go_layout_scene_zaxis_title_font (go_layout_scene), 1006
go_layout_selection, *945*, *1252*
go_layout_selection (add_selection), 562
go_layout_selection_line, *563*
go_layout_selection_line (add_selection), 562
go_layout_shape, *945*, *1253*
go_layout_shape (add_shape), 565
go_layout_shape_label, *569*
go_layout_shape_label (add_shape), 565
go_layout_shape_label_font, *572*
go_layout_shape_label_font (add_shape), 565
go_layout_shape_legendgrouptitle, *569*
go_layout_shape_legendgrouptitle (add_shape), 565
go_layout_shape_legendgrouptitle_font, *572*
go_layout_shape_legendgrouptitle_font (add_shape), 565
go_layout_shape_line, *569*
go_layout_shape_line (add_shape), 565
go_layout_slider, *945*, 1028
go_layout_slider_currentvalue, *1031*
go_layout_slider_currentvalue (go_layout_slider), 1028
go_layout_slider_currentvalue_font, *1031*
go_layout_slider_currentvalue_font (go_layout_slider), 1028
go_layout_slider_font, *1031*
go_layout_slider_font (go_layout_slider), 1028
go_layout_slider_pad, *1031*
go_layout_slider_pad (go_layout_slider), 1028
go_layout_slider_step, *1031*
go_layout_slider_step (go_layout_slider), 1028
go_layout_slider_transition, *1031*
go_layout_slider_transition (go_layout_slider), 1028
go_layout_smith, *945*, 1034, *1254*
go_layout_smith_domain, *1036*
go_layout_smith_domain (go_layout_smith), 1034
go_layout_smith_imaginaryaxis, *1036*
go_layout_smith_imaginaryaxis (go_layout_smith), 1034
go_layout_smith_imaginaryaxis_tickfont, *1038*
go_layout_smith_imaginaryaxis_tickfont (go_layout_smith), 1034
go_layout_smith_realaxis, *1036*

go_layout_smith_realaxis
(go_layout_smith), 1034
go_layout_smith_realaxis_tickfont,
1038
go_layout_smith_realaxis_tickfont
(go_layout_smith), 1034
go_layout_ternary, *946*, 1039, *1255*
go_layout_ternary_aaxis, *1046*
go_layout_ternary_aaxis
(go_layout_ternary), 1039
go_layout_ternary_aaxis_tickfont, *1049*
go_layout_ternary_aaxis_tickfont
(go_layout_ternary), 1039
go_layout_ternary_aaxis_tickformatstop,
1049
go_layout_ternary_aaxis_tickformatstop
(go_layout_ternary), 1039
go_layout_ternary_aaxis_title, *1050*
go_layout_ternary_aaxis_title
(go_layout_ternary), 1039
go_layout_ternary_aaxis_title_font,
1051
go_layout_ternary_aaxis_title_font
(go_layout_ternary), 1039
go_layout_ternary_baxis, *1046*
go_layout_ternary_baxis
(go_layout_ternary), 1039
go_layout_ternary_baxis_tickfont, *1049*
go_layout_ternary_baxis_tickfont
(go_layout_ternary), 1039
go_layout_ternary_baxis_tickformatstop,
1050
go_layout_ternary_baxis_tickformatstop
(go_layout_ternary), 1039
go_layout_ternary_baxis_title, *1050*
go_layout_ternary_baxis_title
(go_layout_ternary), 1039
go_layout_ternary_baxis_title_font,
1051
go_layout_ternary_baxis_title_font
(go_layout_ternary), 1039
go_layout_ternary_caxis, *1046*
go_layout_ternary_caxis
(go_layout_ternary), 1039
go_layout_ternary_caxis_tickfont, *1049*
go_layout_ternary_caxis_tickfont
(go_layout_ternary), 1039
go_layout_ternary_caxis_tickformatstop,
1050
go_layout_ternary_caxis_tickformatstop
(go_layout_ternary), 1039
go_layout_ternary_caxis_title, *1050*
go_layout_ternary_caxis_title
(go_layout_ternary), 1039
go_layout_ternary_caxis_title_font,
1051
go_layout_ternary_caxis_title_font
(go_layout_ternary), 1039
go_layout_ternary_domain, *1046*
go_layout_ternary_domain
(go_layout_ternary), 1039
go_layout_title, *946*, 1053
go_layout_title_font, *1054*
go_layout_title_font (go_layout_title),
1053
go_layout_title_pad, *1055*
go_layout_title_pad (go_layout_title),
1053
go_layout_title_subtitle, *1055*
go_layout_title_subtitle
(go_layout_title), 1053
go_layout_title_subtitle_font, *1054*
go_layout_title_subtitle_font
(go_layout_title), 1053
go_layout_transition, *946*, 1056
go_layout_uniformtext, *946*, 1058
go_layout_updatemenu, *946*, 1059
go_layout_updatemenu_button, *1060*
go_layout_updatemenu_button
(go_layout_updatemenu), 1059
go_layout_updatemenu_font, *1060*
go_layout_updatemenu_font
(go_layout_updatemenu), 1059
go_layout_updatemenu_pad, *1061*
go_layout_updatemenu_pad
(go_layout_updatemenu), 1059
go_layout_xaxis, *947*, 1063, *1256*
go_layout_xaxis_autorangeoptions, *1069*
go_layout_xaxis_autorangeoptions
(go_layout_xaxis), 1063
go_layout_xaxis_domainpad, *1071*
go_layout_xaxis_domainpad
(go_layout_xaxis), 1063
go_layout_xaxis_minor, *1072*
go_layout_xaxis_minor
(go_layout_xaxis), 1063

go_layout_xaxis_rangebreak, *1073*
go_layout_xaxis_rangebreak (go_layout_xaxis), 1063
go_layout_xaxis_rangeselector, *1073*
go_layout_xaxis_rangeselector (go_layout_xaxis), 1063
go_layout_xaxis_rangeselector_button, *1079*
go_layout_xaxis_rangeselector_button (go_layout_xaxis), 1063
go_layout_xaxis_rangeselector_font, *1079*
go_layout_xaxis_rangeselector_font (go_layout_xaxis), 1063
go_layout_xaxis_rangeslider, *1073*
go_layout_xaxis_rangeslider (go_layout_xaxis), 1063
go_layout_xaxis_rangeslider_yaxis, *1080*
go_layout_xaxis_rangeslider_yaxis (go_layout_xaxis), 1063
go_layout_xaxis_tickfont, *1075*
go_layout_xaxis_tickfont (go_layout_xaxis), 1063
go_layout_xaxis_tickformatstop, *1075*
go_layout_xaxis_tickformatstop (go_layout_xaxis), 1063
go_layout_xaxis_title, *1077*
go_layout_xaxis_title (go_layout_xaxis), 1063
go_layout_xaxis_title_font, *1079*
go_layout_xaxis_title_font (go_layout_xaxis), 1063
go_layout_xaxis_unifiedhovertitle, *1077*
go_layout_xaxis_unifiedhovertitle (go_layout_xaxis), 1063
go_layout_yaxis, *947*, 1081, *1257*, *1258*
go_layout_yaxis_autorangeoptions, *1086*
go_layout_yaxis_autorangeoptions (go_layout_yaxis), 1081
go_layout_yaxis_domainpad, *1088*
go_layout_yaxis_domainpad (go_layout_yaxis), 1081
go_layout_yaxis_minor, *1089*
go_layout_yaxis_minor (go_layout_yaxis), 1081
go_layout_yaxis_rangebreak, *1090*
go_layout_yaxis_rangebreak (go_layout_yaxis), 1081
go_layout_yaxis_tickfont, *1092*
go_layout_yaxis_tickfont (go_layout_yaxis), 1081
go_layout_yaxis_tickformatstop, *1092*
go_layout_yaxis_tickformatstop (go_layout_yaxis), 1081
go_layout_yaxis_title, *1094*
go_layout_yaxis_title (go_layout_yaxis), 1081
go_layout_yaxis_title_font, *1096*
go_layout_yaxis_title_font (go_layout_yaxis), 1081
go_layout_yaxis_unifiedhovertitle, *1094*
go_layout_yaxis_unifiedhovertitle (go_layout_yaxis), 1081
go_mesh3d (add_mesh3d), 207
go_mesh3d_colorbar, *215*
go_mesh3d_colorbar (add_mesh3d), 207
go_mesh3d_colorbar_tickfont, *222*
go_mesh3d_colorbar_tickfont (add_mesh3d), 207
go_mesh3d_colorbar_tickformatstop, *222*
go_mesh3d_colorbar_tickformatstop (add_mesh3d), 207
go_mesh3d_colorbar_title, *223*
go_mesh3d_colorbar_title (add_mesh3d), 207
go_mesh3d_colorbar_title_font, *225*
go_mesh3d_colorbar_title_font (add_mesh3d), 207
go_mesh3d_contour, *215*
go_mesh3d_contour (add_mesh3d), 207
go_mesh3d_hoverlabel, *216*
go_mesh3d_hoverlabel (add_mesh3d), 207
go_mesh3d_hoverlabel_font, *225*
go_mesh3d_hoverlabel_font (add_mesh3d), 207
go_mesh3d_legendgrouptitle, *217*
go_mesh3d_legendgrouptitle (add_mesh3d), 207
go_mesh3d_legendgrouptitle_font, *225*
go_mesh3d_legendgrouptitle_font (add_mesh3d), 207
go_mesh3d_lighting, *217*
go_mesh3d_lighting (add_mesh3d), 207

go_mesh3d_lightposition, *217*
go_mesh3d_lightposition *(*add_mesh3d*)*, *207*
go_ohlc *(*add_ohlc*)*, 226
go_ohlc_decreasing, *230*
go_ohlc_decreasing *(*add_ohlc*)*, 226
go_ohlc_decreasing_line, *232*
go_ohlc_decreasing_line *(*add_ohlc*)*, 226
go_ohlc_hoverlabel, *231*
go_ohlc_hoverlabel *(*add_ohlc*)*, 226
go_ohlc_hoverlabel_font, *235*
go_ohlc_hoverlabel_font *(*add_ohlc*)*, 226
go_ohlc_increasing, *231*
go_ohlc_increasing *(*add_ohlc*)*, 226
go_ohlc_increasing_line, *232*
go_ohlc_increasing_line *(*add_ohlc*)*, 226
go_ohlc_legendgrouptitle, *231*
go_ohlc_legendgrouptitle *(*add_ohlc*)*, 226
go_ohlc_legendgrouptitle_font, *235*
go_ohlc_legendgrouptitle_font *(*add_ohlc*)*, 226
go_ohlc_line, *232*
go_ohlc_line *(*add_ohlc*)*, 226
go_parcats *(*add_parcats*)*, 237
go_parcats_dimension, *243*
go_parcats_dimension *(*add_parcats*)*, 237
go_parcats_domain, *243*
go_parcats_domain *(*add_parcats*)*, 237
go_parcats_labelfont, *244*
go_parcats_labelfont *(*add_parcats*)*, 237
go_parcats_legendgrouptitle, *244*
go_parcats_legendgrouptitle *(*add_parcats*)*, 237
go_parcats_legendgrouptitle_font, *247*
go_parcats_legendgrouptitle_font *(*add_parcats*)*, 237
go_parcats_line, *244*
go_parcats_line *(*add_parcats*)*, 237
go_parcats_line_colorbar, *248*
go_parcats_line_colorbar *(*add_parcats*)*, 237
go_parcats_line_colorbar_tickfont, *245*
go_parcats_line_colorbar_tickfont *(*add_parcats*)*, 237
go_parcats_line_colorbar_tickformatstop, *250*
go_parcats_line_colorbar_tickformatstop *(*add_parcats*)*, 237
go_parcats_line_colorbar_title, *250*
go_parcats_line_colorbar_title *(*add_parcats*)*, 237
go_parcats_line_colorbar_title_font, *247*
go_parcats_line_colorbar_title_font *(*add_parcats*)*, 237
go_parcats_tickfont, *245*
go_parcats_tickfont *(*add_parcats*)*, 237
go_parcoords *(*add_parcoords*)*, 252
go_parcoords_dimension, *258*
go_parcoords_dimension *(*add_parcoords*)*, 252
go_parcoords_domain, *258*
go_parcoords_domain *(*add_parcoords*)*, 252
go_parcoords_labelfont, *259*
go_parcoords_labelfont *(*add_parcoords*)*, 252
go_parcoords_legendgrouptitle, *259*
go_parcoords_legendgrouptitle *(*add_parcoords*)*, 252
go_parcoords_legendgrouptitle_font, *262*
go_parcoords_legendgrouptitle_font *(*add_parcoords*)*, 252
go_parcoords_line, *259*
go_parcoords_line *(*add_parcoords*)*, 252
go_parcoords_line_colorbar, *263*
go_parcoords_line_colorbar *(*add_parcoords*)*, 252
go_parcoords_line_colorbar_tickfont, *260*
go_parcoords_line_colorbar_tickfont *(*add_parcoords*)*, 252
go_parcoords_line_colorbar_tickformatstop, *265*
go_parcoords_line_colorbar_tickformatstop *(*add_parcoords*)*, 252
go_parcoords_line_colorbar_title, *265*
go_parcoords_line_colorbar_title *(*add_parcoords*)*, 252
go_parcoords_line_colorbar_title_font, *262*
go_parcoords_line_colorbar_title_font *(*add_parcoords*)*, 252
go_parcoords_rangefont, *259*
go_parcoords_rangefont *(*add_parcoords*)*, 252

go_parcoords_tickfont, *260*
go_parcoords_tickfont *(*add_parcoords*)*, 252
go_parcoords_unselected, *260*
go_parcoords_unselected *(*add_parcoords*)*, 252
go_parcoords_unselected_line, *259*
go_parcoords_unselected_line *(*add_parcoords*)*, 252
go_pie *(*add_pie*)*, 267
go_pie_domain, *273*
go_pie_domain *(*add_pie*)*, 267
go_pie_hoverlabel, *273*
go_pie_hoverlabel *(*add_pie*)*, 267
go_pie_hoverlabel_font, *277*
go_pie_hoverlabel_font *(*add_pie*)*, 267
go_pie_insidetextfont, *274*
go_pie_insidetextfont *(*add_pie*)*, 267
go_pie_legendgrouptitle, *275*
go_pie_legendgrouptitle *(*add_pie*)*, 267
go_pie_legendgrouptitle_font, *277*
go_pie_legendgrouptitle_font *(*add_pie*)*, 267
go_pie_marker, *275*
go_pie_marker *(*add_pie*)*, 267
go_pie_marker_line, *279*
go_pie_marker_line *(*add_pie*)*, 267
go_pie_marker_pattern, *279*
go_pie_marker_pattern *(*add_pie*)*, 267
go_pie_outsidetextfont, *275*
go_pie_outsidetextfont *(*add_pie*)*, 267
go_pie_textfont, *276*
go_pie_textfont *(*add_pie*)*, 267
go_pie_title, *276*
go_pie_title *(*add_pie*)*, 267
go_pie_title_font, *277*
go_pie_title_font *(*add_pie*)*, 267
go_quiver *(*add_quiver*)*, 281
go_quiver_hoverlabel, *289*
go_quiver_hoverlabel *(*add_quiver*)*, 281
go_quiver_hoverlabel_font, *293*
go_quiver_hoverlabel_font *(*add_quiver*)*, 281
go_quiver_legendgrouptitle, *290*
go_quiver_legendgrouptitle *(*add_quiver*)*, 281
go_quiver_legendgrouptitle_font, *293*
go_quiver_legendgrouptitle_font *(*add_quiver*)*, 281
go_quiver_marker, *290*
go_quiver_marker *(*add_quiver*)*, 281
go_quiver_marker_colorbar, *296*
go_quiver_marker_colorbar *(*add_quiver*)*, 281
go_quiver_marker_colorbar_tickfont, *298*
go_quiver_marker_colorbar_tickfont *(*add_quiver*)*, 281
go_quiver_marker_colorbar_tickformatstop, *298*
go_quiver_marker_colorbar_tickformatstop *(*add_quiver*)*, 281
go_quiver_marker_colorbar_title, *299*
go_quiver_marker_colorbar_title *(*add_quiver*)*, 281
go_quiver_marker_colorbar_title_font, *293*
go_quiver_marker_colorbar_title_font *(*add_quiver*)*, 281
go_quiver_marker_line, *296*
go_quiver_marker_line *(*add_quiver*)*, 281
go_quiver_selected, *290*
go_quiver_selected *(*add_quiver*)*, 281
go_quiver_selected_marker, *290*
go_quiver_selected_marker *(*add_quiver*)*, 281
go_quiver_selected_marker_line, *296*
go_quiver_selected_marker_line *(*add_quiver*)*, 281
go_quiver_selected_textfont, *291*
go_quiver_selected_textfont *(*add_quiver*)*, 281
go_quiver_textfont, *291*
go_quiver_textfont *(*add_quiver*)*, 281
go_quiver_unselected, *291*
go_quiver_unselected *(*add_quiver*)*, 281
go_quiver_unselected_marker, *290*
go_quiver_unselected_marker *(*add_quiver*)*, 281
go_quiver_unselected_marker_line, *296*
go_quiver_unselected_marker_line *(*add_quiver*)*, 281
go_quiver_unselected_textfont, *291*
go_quiver_unselected_textfont *(*add_quiver*)*, 281
go_sankey *(*add_sankey*)*, 301

go_sankey_domain, *307*
go_sankey_domain *(*add_sankey*)*, 301
go_sankey_hoverlabel, *307*
go_sankey_hoverlabel *(*add_sankey*)*, 301
go_sankey_hoverlabel_font, *310*
go_sankey_hoverlabel_font *(*add_sankey*)*, 301
go_sankey_legendgrouptitle, *308*
go_sankey_legendgrouptitle *(*add_sankey*)*, 301
go_sankey_legendgrouptitle_font, *310*
go_sankey_legendgrouptitle_font *(*add_sankey*)*, 301
go_sankey_link, *308*
go_sankey_link *(*add_sankey*)*, 301
go_sankey_link_concentrationscales, *312*
go_sankey_link_concentrationscales *(*add_sankey*)*, 301
go_sankey_link_hoverlabel, *307*
go_sankey_link_hoverlabel *(*add_sankey*)*, 301
go_sankey_link_hoverlabel_font, *310*
go_sankey_link_hoverlabel_font *(*add_sankey*)*, 301
go_sankey_link_line, *312*
go_sankey_link_line *(*add_sankey*)*, 301
go_sankey_node, *308*
go_sankey_node *(*add_sankey*)*, 301
go_sankey_node_hoverlabel, *307*
go_sankey_node_hoverlabel *(*add_sankey*)*, 301
go_sankey_node_hoverlabel_font, *310*
go_sankey_node_hoverlabel_font *(*add_sankey*)*, 301
go_sankey_node_line, *312*
go_sankey_node_line *(*add_sankey*)*, 301
go_sankey_textfont, *308*
go_sankey_textfont *(*add_sankey*)*, 301
go_scatter *(*add_scatter*)*, 314
go_scatter(), *664*, *937*
go_scatter3d *(*add_scatter3d*)*, 345
go_scatter3d_error_x, *356*
go_scatter3d_error_x *(*add_scatter3d*)*, 345
go_scatter3d_error_y, *356*
go_scatter3d_error_y *(*add_scatter3d*)*, 345
go_scatter3d_error_z, *356*
go_scatter3d_error_z *(*add_scatter3d*)*, 345
go_scatter3d_hoverlabel, *356*
go_scatter3d_hoverlabel *(*add_scatter3d*)*, 345
go_scatter3d_hoverlabel_font, *363*
go_scatter3d_hoverlabel_font *(*add_scatter3d*)*, 345
go_scatter3d_legendgrouptitle, *357*
go_scatter3d_legendgrouptitle *(*add_scatter3d*)*, 345
go_scatter3d_legendgrouptitle_font, *363*
go_scatter3d_legendgrouptitle_font *(*add_scatter3d*)*, 345
go_scatter3d_line, *358*
go_scatter3d_line *(*add_scatter3d*)*, 345
go_scatter3d_line_colorbar, *366*
go_scatter3d_line_colorbar *(*add_scatter3d*)*, 345
go_scatter3d_line_colorbar_tickfont, *369*
go_scatter3d_line_colorbar_tickfont *(*add_scatter3d*)*, 345
go_scatter3d_line_colorbar_tickformatstop, *369*
go_scatter3d_line_colorbar_tickformatstop *(*add_scatter3d*)*, 345
go_scatter3d_line_colorbar_title, *370*
go_scatter3d_line_colorbar_title *(*add_scatter3d*)*, 345
go_scatter3d_line_colorbar_title_font, *363*
go_scatter3d_line_colorbar_title_font *(*add_scatter3d*)*, 345
go_scatter3d_marker, *358*
go_scatter3d_marker *(*add_scatter3d*)*, 345
go_scatter3d_marker_colorbar, *366*
go_scatter3d_marker_colorbar *(*add_scatter3d*)*, 345
go_scatter3d_marker_colorbar_tickfont, *369*
go_scatter3d_marker_colorbar_tickfont *(*add_scatter3d*)*, 345
go_scatter3d_marker_colorbar_tickformatstop, *369*
go_scatter3d_marker_colorbar_tickformatstop

(add_scatter3d), 345
go_scatter3d_marker_colorbar_title, 370
go_scatter3d_marker_colorbar_title (add_scatter3d), 345
go_scatter3d_marker_colorbar_title_font, 363
go_scatter3d_marker_colorbar_title_font (add_scatter3d), 345
go_scatter3d_marker_line, 358
go_scatter3d_marker_line (add_scatter3d), 345
go_scatter3d_projection, 358
go_scatter3d_projection (add_scatter3d), 345
go_scatter3d_projection_x, 360
go_scatter3d_projection_x (add_scatter3d), 345
go_scatter3d_projection_y, 360
go_scatter3d_projection_y (add_scatter3d), 345
go_scatter3d_projection_z, 361
go_scatter3d_projection_z (add_scatter3d), 345
go_scatter3d_textfont, 359
go_scatter3d_textfont (add_scatter3d), 345
go_scatter_error_x, 325
go_scatter_error_x (add_scatter), 314
go_scatter_error_y, 325
go_scatter_error_y (add_scatter), 314
go_scatter_fillgradient, 326
go_scatter_fillgradient (add_scatter), 314
go_scatter_fillpattern, 326
go_scatter_fillpattern (add_scatter), 314
go_scatter_hoverlabel, 326
go_scatter_hoverlabel (add_scatter), 314
go_scatter_hoverlabel_font, 335
go_scatter_hoverlabel_font (add_scatter), 314
go_scatter_legendgrouptitle, 327
go_scatter_legendgrouptitle (add_scatter), 314
go_scatter_legendgrouptitle_font, 335
go_scatter_legendgrouptitle_font (add_scatter), 314
go_scatter_line, 327
go_scatter_line (add_scatter), 314
go_scatter_marker, 327
go_scatter_marker (add_scatter), 314
go_scatter_marker_colorbar, 339
go_scatter_marker_colorbar (add_scatter), 314
go_scatter_marker_colorbar_tickfont, 342
go_scatter_marker_colorbar_tickfont (add_scatter), 314
go_scatter_marker_colorbar_tickformatstop, 342
go_scatter_marker_colorbar_tickformatstop (add_scatter), 314
go_scatter_marker_colorbar_title, 343
go_scatter_marker_colorbar_title (add_scatter), 314
go_scatter_marker_colorbar_title_font, 336
go_scatter_marker_colorbar_title_font (add_scatter), 314
go_scatter_marker_gradient, 339
go_scatter_marker_gradient (add_scatter), 314
go_scatter_marker_line, 327
go_scatter_marker_line (add_scatter), 314
go_scatter_selected, 328
go_scatter_selected (add_scatter), 314
go_scatter_selected_marker, 327
go_scatter_selected_marker (add_scatter), 314
go_scatter_selected_textfont, 329
go_scatter_selected_textfont (add_scatter), 314
go_scatter_textfont, 329
go_scatter_textfont (add_scatter), 314
go_scatter_unselected, 330
go_scatter_unselected (add_scatter), 314
go_scatter_unselected_marker, 327
go_scatter_unselected_marker (add_scatter), 314
go_scatter_unselected_textfont, 329
go_scatter_unselected_textfont (add_scatter), 314
go_scattercarpet (add_scattercarpet), 372

go_scattercarpet_hoverlabel, *381*
go_scattercarpet_hoverlabel
(add_scattercarpet), 372
go_scattercarpet_hoverlabel_font, *385*
go_scattercarpet_hoverlabel_font
(add_scattercarpet), 372
go_scattercarpet_legendgrouptitle, *382*
go_scattercarpet_legendgrouptitle
(add_scattercarpet), 372
go_scattercarpet_legendgrouptitle_font,
385
go_scattercarpet_legendgrouptitle_font
(add_scattercarpet), 372
go_scattercarpet_line, *382*
go_scattercarpet_line
(add_scattercarpet), 372
go_scattercarpet_marker, *383*
go_scattercarpet_marker
(add_scattercarpet), 372
go_scattercarpet_marker_colorbar, *390*
go_scattercarpet_marker_colorbar
(add_scattercarpet), 372
go_scattercarpet_marker_colorbar_tickfont,
393
go_scattercarpet_marker_colorbar_tickfont
(add_scattercarpet), 372
go_scattercarpet_marker_colorbar_tickformatstop,
394
go_scattercarpet_marker_colorbar_tickformatstop
(add_scattercarpet), 372
go_scattercarpet_marker_colorbar_title,
394
go_scattercarpet_marker_colorbar_title
(add_scattercarpet), 372
go_scattercarpet_marker_colorbar_title_font,
385
go_scattercarpet_marker_colorbar_title_font
(add_scattercarpet), 372
go_scattercarpet_marker_gradient, *390*
go_scattercarpet_marker_gradient
(add_scattercarpet), 372
go_scattercarpet_marker_line, *382*
go_scattercarpet_marker_line
(add_scattercarpet), 372
go_scattercarpet_selected, *383*
go_scattercarpet_selected
(add_scattercarpet), 372
go_scattercarpet_selected_marker, *383*
go_scattercarpet_selected_marker
(add_scattercarpet), 372
go_scattercarpet_selected_textfont,
384
go_scattercarpet_selected_textfont
(add_scattercarpet), 372
go_scattercarpet_textfont, *384*
go_scattercarpet_textfont
(add_scattercarpet), 372
go_scattercarpet_unselected, *385*
go_scattercarpet_unselected
(add_scattercarpet), 372
go_scattercarpet_unselected_marker,
383
go_scattercarpet_unselected_marker
(add_scattercarpet), 372
go_scattercarpet_unselected_textfont,
384
go_scattercarpet_unselected_textfont
(add_scattercarpet), 372
go_scattergeo *(*add_scattergeo*)*, 397
go_scattergeo_hoverlabel, *405*
go_scattergeo_hoverlabel
(add_scattergeo), 397
go_scattergeo_hoverlabel_font, *409*
go_scattergeo_hoverlabel_font
(add_scattergeo), 397
go_scattergeo_legendgrouptitle, *406*
go_scattergeo_legendgrouptitle
(add_scattergeo), 397
go_scattergeo_legendgrouptitle_font,
409
go_scattergeo_legendgrouptitle_font
(add_scattergeo), 397
go_scattergeo_line, *407*
go_scattergeo_line *(*add_scattergeo*)*, 397
go_scattergeo_marker, *407*
go_scattergeo_marker *(*add_scattergeo*)*,
397
go_scattergeo_marker_colorbar, *413*
go_scattergeo_marker_colorbar
(add_scattergeo), 397
go_scattergeo_marker_colorbar_tickfont,
417
go_scattergeo_marker_colorbar_tickfont
(add_scattergeo), 397
go_scattergeo_marker_colorbar_tickformatstop,
417

go_scattergeo_marker_colorbar_tickformatstop (add_scattergeo), 397
go_scattergeo_marker_colorbar_title, 418
go_scattergeo_marker_colorbar_title (add_scattergeo), 397
go_scattergeo_marker_colorbar_title_font, 409
go_scattergeo_marker_colorbar_title_font (add_scattergeo), 397
go_scattergeo_marker_gradient, 414
go_scattergeo_marker_gradient (add_scattergeo), 397
go_scattergeo_marker_line, 407
go_scattergeo_marker_line (add_scattergeo), 397
go_scattergeo_selected, 408
go_scattergeo_selected (add_scattergeo), 397
go_scattergeo_selected_marker, 407
go_scattergeo_selected_marker (add_scattergeo), 397
go_scattergeo_selected_textfont, 408
go_scattergeo_selected_textfont (add_scattergeo), 397
go_scattergeo_textfont, 408
go_scattergeo_textfont (add_scattergeo), 397
go_scattergeo_unselected, 409
go_scattergeo_unselected (add_scattergeo), 397
go_scattergeo_unselected_marker, 407
go_scattergeo_unselected_marker (add_scattergeo), 397
go_scattergeo_unselected_textfont, 408
go_scattergeo_unselected_textfont (add_scattergeo), 397
go_scattergl (add_scattergl), 420
go_scattergl(), 740
go_scattergl_error_x, 430
go_scattergl_error_x (add_scattergl), 420
go_scattergl_error_y, 430
go_scattergl_error_y (add_scattergl), 420
go_scattergl_hoverlabel, 430
go_scattergl_hoverlabel (add_scattergl), 420
go_scattergl_hoverlabel_font, 437
go_scattergl_hoverlabel_font (add_scattergl), 420
go_scattergl_legendgrouptitle, 431
go_scattergl_legendgrouptitle (add_scattergl), 420
go_scattergl_legendgrouptitle_font, 437
go_scattergl_legendgrouptitle_font (add_scattergl), 420
go_scattergl_line, 431
go_scattergl_line (add_scattergl), 420
go_scattergl_marker, 432
go_scattergl_marker (add_scattergl), 420
go_scattergl_marker_colorbar, 440
go_scattergl_marker_colorbar (add_scattergl), 420
go_scattergl_marker_colorbar_tickfont, 444
go_scattergl_marker_colorbar_tickfont (add_scattergl), 420
go_scattergl_marker_colorbar_tickformatstop, 444
go_scattergl_marker_colorbar_tickformatstop (add_scattergl), 420
go_scattergl_marker_colorbar_title, 445
go_scattergl_marker_colorbar_title (add_scattergl), 420
go_scattergl_marker_colorbar_title_font, 437
go_scattergl_marker_colorbar_title_font (add_scattergl), 420
go_scattergl_marker_line, 431
go_scattergl_marker_line (add_scattergl), 420
go_scattergl_selected, 432
go_scattergl_selected (add_scattergl), 420
go_scattergl_selected_marker, 432
go_scattergl_selected_marker (add_scattergl), 420
go_scattergl_selected_textfont, 433
go_scattergl_selected_textfont (add_scattergl), 420
go_scattergl_textfont, 433
go_scattergl_textfont (add_scattergl), 420

go_scattergl_unselected, *433*
go_scattergl_unselected (add_scattergl), 420
go_scattergl_unselected_marker, *432*
go_scattergl_unselected_marker (add_scattergl), 420
go_scattergl_unselected_textfont, *433*
go_scattergl_unselected_textfont (add_scattergl), 420
go_scattermap (add_scattermap), 447
go_scattermap_cluster, *454*
go_scattermap_cluster (add_scattermap), 447
go_scattermap_hoverlabel, *454*
go_scattermap_hoverlabel (add_scattermap), 447
go_scattermap_hoverlabel_font, *459*
go_scattermap_hoverlabel_font (add_scattermap), 447
go_scattermap_legendgrouptitle, *455*
go_scattermap_legendgrouptitle (add_scattermap), 447
go_scattermap_legendgrouptitle_font, *459*
go_scattermap_legendgrouptitle_font (add_scattermap), 447
go_scattermap_line, *455*
go_scattermap_line (add_scattermap), 447
go_scattermap_marker, *455*
go_scattermap_marker (add_scattermap), 447
go_scattermap_marker_colorbar, *461*
go_scattermap_marker_colorbar (add_scattermap), 447
go_scattermap_marker_colorbar_tickfont, *463*
go_scattermap_marker_colorbar_tickfont (add_scattermap), 447
go_scattermap_marker_colorbar_tickformatstop, *463*
go_scattermap_marker_colorbar_tickformatstop (add_scattermap), 447
go_scattermap_marker_colorbar_title, *464*
go_scattermap_marker_colorbar_title (add_scattermap), 447
go_scattermap_marker_colorbar_title_font, *459*
go_scattermap_marker_colorbar_title_font (add_scattermap), 447
go_scattermap_selected, *456*
go_scattermap_selected (add_scattermap), 447
go_scattermap_selected_marker, *455*
go_scattermap_selected_marker (add_scattermap), 447
go_scattermap_textfont, *457*
go_scattermap_textfont (add_scattermap), 447
go_scattermap_unselected, *457*
go_scattermap_unselected (add_scattermap), 447
go_scattermap_unselected_marker, *455*
go_scattermap_unselected_marker (add_scattermap), 447
go_scatterpolar (add_scatterpolar), 466
go_scatterpolar_hoverlabel, *475*
go_scatterpolar_hoverlabel (add_scatterpolar), 466
go_scatterpolar_hoverlabel_font, *479*
go_scatterpolar_hoverlabel_font (add_scatterpolar), 466
go_scatterpolar_legendgrouptitle, *476*
go_scatterpolar_legendgrouptitle (add_scatterpolar), 466
go_scatterpolar_legendgrouptitle_font, *479*
go_scatterpolar_legendgrouptitle_font (add_scatterpolar), 466
go_scatterpolar_line, *476*
go_scatterpolar_line (add_scatterpolar), 466
go_scatterpolar_marker, *476*
go_scatterpolar_marker (add_scatterpolar), 466
go_scatterpolar_marker_colorbar, *483*
go_scatterpolar_marker_colorbar (add_scatterpolar), 466
go_scatterpolar_marker_colorbar_tickfont, *487*
go_scatterpolar_marker_colorbar_tickfont (add_scatterpolar), 466
go_scatterpolar_marker_colorbar_tickformatstop, *487*
go_scatterpolar_marker_colorbar_tickformatstop (add_scatterpolar), 466

go_scatterpolar_marker_colorbar_title, *488*
go_scatterpolar_marker_colorbar_title (add_scatterpolar), 466
go_scatterpolar_marker_colorbar_title_font, *479*
go_scatterpolar_marker_colorbar_title_font (add_scatterpolar), 466
go_scatterpolar_marker_gradient, *484*
go_scatterpolar_marker_gradient (add_scatterpolar), 466
go_scatterpolar_marker_line, *476*
go_scatterpolar_marker_line (add_scatterpolar), 466
go_scatterpolar_selected, *477*
go_scatterpolar_selected (add_scatterpolar), 466
go_scatterpolar_selected_marker, *476*
go_scatterpolar_selected_marker (add_scatterpolar), 466
go_scatterpolar_selected_textfont, *478*
go_scatterpolar_selected_textfont (add_scatterpolar), 466
go_scatterpolar_textfont, *478*
go_scatterpolar_textfont (add_scatterpolar), 466
go_scatterpolar_unselected, *479*
go_scatterpolar_unselected (add_scatterpolar), 466
go_scatterpolar_unselected_marker, *476*
go_scatterpolar_unselected_marker (add_scatterpolar), 466
go_scatterpolar_unselected_textfont, *478*
go_scatterpolar_unselected_textfont (add_scatterpolar), 466
go_scatterpolargl (add_scatterpolargl), 490
go_scatterpolargl_hoverlabel, *499*
go_scatterpolargl_hoverlabel (add_scatterpolargl), 490
go_scatterpolargl_hoverlabel_font, *503*
go_scatterpolargl_hoverlabel_font (add_scatterpolargl), 490
go_scatterpolargl_legendgrouptitle, *500*
go_scatterpolargl_legendgrouptitle (add_scatterpolargl), 490
go_scatterpolargl_legendgrouptitle_font, *503*
go_scatterpolargl_legendgrouptitle_font (add_scatterpolargl), 490
go_scatterpolargl_line, *500*
go_scatterpolargl_line (add_scatterpolargl), 490
go_scatterpolargl_marker, *500*
go_scatterpolargl_marker (add_scatterpolargl), 490
go_scatterpolargl_marker_colorbar, *507*
go_scatterpolargl_marker_colorbar (add_scatterpolargl), 490
go_scatterpolargl_marker_colorbar_tickfont, *510*
go_scatterpolargl_marker_colorbar_tickfont (add_scatterpolargl), 490
go_scatterpolargl_marker_colorbar_tickformatstop, *511*
go_scatterpolargl_marker_colorbar_tickformatstop (add_scatterpolargl), 490
go_scatterpolargl_marker_colorbar_title, *511*
go_scatterpolargl_marker_colorbar_title (add_scatterpolargl), 490
go_scatterpolargl_marker_colorbar_title_font, *503*
go_scatterpolargl_marker_colorbar_title_font (add_scatterpolargl), 490
go_scatterpolargl_marker_line, *500*
go_scatterpolargl_marker_line (add_scatterpolargl), 490
go_scatterpolargl_selected, *501*
go_scatterpolargl_selected (add_scatterpolargl), 490
go_scatterpolargl_selected_marker, *500*
go_scatterpolargl_selected_marker (add_scatterpolargl), 490
go_scatterpolargl_selected_textfont, *502*
go_scatterpolargl_selected_textfont (add_scatterpolargl), 490
go_scatterpolargl_textfont, *501*
go_scatterpolargl_textfont (add_scatterpolargl), 490
go_scatterpolargl_unselected, *502*
go_scatterpolargl_unselected (add_scatterpolargl), 490

go_scatterpolargl_unselected_marker, *500*
go_scatterpolargl_unselected_marker *(*add_scatterpolargl*)*, 490
go_scatterpolargl_unselected_textfont, *502*
go_scatterpolargl_unselected_textfont *(*add_scatterpolargl*)*, 490
go_scattersmith *(*add_scattersmith*)*, 514
go_scattersmith_hoverlabel, *522*
go_scattersmith_hoverlabel *(*add_scattersmith*)*, 514
go_scattersmith_hoverlabel_font, *526*
go_scattersmith_hoverlabel_font *(*add_scattersmith*)*, 514
go_scattersmith_legendgrouptitle, *523*
go_scattersmith_legendgrouptitle *(*add_scattersmith*)*, 514
go_scattersmith_legendgrouptitle_font, *526*
go_scattersmith_legendgrouptitle_font *(*add_scattersmith*)*, 514
go_scattersmith_line, *524*
go_scattersmith_line *(*add_scattersmith*)*, 514
go_scattersmith_marker, *524*
go_scattersmith_marker *(*add_scattersmith*)*, 514
go_scattersmith_marker_colorbar, *531*
go_scattersmith_marker_colorbar *(*add_scattersmith*)*, 514
go_scattersmith_marker_colorbar_tickfont, *534*
go_scattersmith_marker_colorbar_tickfont *(*add_scattersmith*)*, 514
go_scattersmith_marker_colorbar_tickformatstop, *535*
go_scattersmith_marker_colorbar_tickformatstop *(*add_scattersmith*)*, 514
go_scattersmith_marker_colorbar_title, *535*
go_scattersmith_marker_colorbar_title *(*add_scattersmith*)*, 514
go_scattersmith_marker_colorbar_title_font, *526*
go_scattersmith_marker_colorbar_title_font *(*add_scattersmith*)*, 514
go_scattersmith_marker_gradient, *531*
go_scattersmith_marker_gradient *(*add_scattersmith*)*, 514
go_scattersmith_marker_line, *524*
go_scattersmith_marker_line *(*add_scattersmith*)*, 514
go_scattersmith_selected, *524*
go_scattersmith_selected *(*add_scattersmith*)*, 514
go_scattersmith_selected_marker, *524*
go_scattersmith_selected_marker *(*add_scattersmith*)*, 514
go_scattersmith_selected_textfont, *525*
go_scattersmith_selected_textfont *(*add_scattersmith*)*, 514
go_scattersmith_textfont, *525*
go_scattersmith_textfont *(*add_scattersmith*)*, 514
go_scattersmith_unselected, *526*
go_scattersmith_unselected *(*add_scattersmith*)*, 514
go_scattersmith_unselected_marker, *524*
go_scattersmith_unselected_marker *(*add_scattersmith*)*, 514
go_scattersmith_unselected_textfont, *525*
go_scattersmith_unselected_textfont *(*add_scattersmith*)*, 514
go_scatterternary *(*add_scatterternary*)*, 538
go_scatterternary_hoverlabel, *547*
go_scatterternary_hoverlabel *(*add_scatterternary*)*, 538
go_scatterternary_hoverlabel_font, *551*
go_scatterternary_hoverlabel_font *(*add_scatterternary*)*, 538
go_scatterternary_legendgrouptitle, *548*
go_scatterternary_legendgrouptitle *(*add_scatterternary*)*, 538
go_scatterternary_legendgrouptitle_font, *551*
go_scatterternary_legendgrouptitle_font *(*add_scatterternary*)*, 538
go_scatterternary_line, *548*
go_scatterternary_line *(*add_scatterternary*)*, 538
go_scatterternary_marker, *548*
go_scatterternary_marker

(add_scatterternary), 538
go_scatterternary_marker_colorbar, *555*
go_scatterternary_marker_colorbar (add_scatterternary), 538
go_scatterternary_marker_colorbar_tickfont, *559*
go_scatterternary_marker_colorbar_tickfont (add_scatterternary), 538
go_scatterternary_marker_colorbar_tickformatstop, *559*
go_scatterternary_marker_colorbar_tickformatstop (add_scatterternary), 538
go_scatterternary_marker_colorbar_title, *560*
go_scatterternary_marker_colorbar_title (add_scatterternary), 538
go_scatterternary_marker_colorbar_title_font, *551*
go_scatterternary_marker_colorbar_title_font (add_scatterternary), 538
go_scatterternary_marker_gradient, *555*
go_scatterternary_marker_gradient (add_scatterternary), 538
go_scatterternary_marker_line, *548*
go_scatterternary_marker_line (add_scatterternary), 538
go_scatterternary_selected, *549*
go_scatterternary_selected (add_scatterternary), 538
go_scatterternary_selected_marker, *548*
go_scatterternary_selected_marker (add_scatterternary), 538
go_scatterternary_selected_textfont, *549*
go_scatterternary_selected_textfont (add_scatterternary), 538
go_scatterternary_textfont, *549*
go_scatterternary_textfont (add_scatterternary), 538
go_scatterternary_unselected, *550*
go_scatterternary_unselected (add_scatterternary), 538
go_scatterternary_unselected_marker, *548*
go_scatterternary_unselected_marker (add_scatterternary), 538
go_scatterternary_unselected_textfont, *549*
go_scatterternary_unselected_textfont (add_scatterternary), 538
go_splom (add_splom), 574
go_splom_diagonal, *581*
go_splom_diagonal (add_splom), 574
go_splom_dimension, *581*
go_splom_dimension (add_splom), 574
go_splom_dimension_axis, *585*
go_splom_dimension_axis (add_splom), 574
go_splom_hoverlabel, *581*
go_splom_hoverlabel (add_splom), 574
go_splom_hoverlabel_font, *585*
go_splom_hoverlabel_font (add_splom), 574
go_splom_legendgrouptitle, *582*
go_splom_legendgrouptitle (add_splom), 574
go_splom_legendgrouptitle_font, *585*
go_splom_legendgrouptitle_font (add_splom), 574
go_splom_marker, *582*
go_splom_marker (add_splom), 574
go_splom_marker_colorbar, *588*
go_splom_marker_colorbar (add_splom), 574
go_splom_marker_colorbar_tickfont, *592*
go_splom_marker_colorbar_tickfont (add_splom), 574
go_splom_marker_colorbar_tickformatstop, *592*
go_splom_marker_colorbar_tickformatstop (add_splom), 574
go_splom_marker_colorbar_title, *593*
go_splom_marker_colorbar_title (add_splom), 574
go_splom_marker_colorbar_title_font, *585*
go_splom_marker_colorbar_title_font (add_splom), 574
go_splom_marker_line, *589*
go_splom_marker_line (add_splom), 574
go_splom_selected, *583*
go_splom_selected (add_splom), 574
go_splom_selected_marker, *582*
go_splom_selected_marker (add_splom), 574
go_splom_unselected, *584*
go_splom_unselected (add_splom), 574

go_splom_unselected_marker, *582*
go_splom_unselected_marker *(*add_splom*)*, 574
go_streamtube *(*add_streamtube*)*, 595
go_streamtube_colorbar, *602*
go_streamtube_colorbar *(*add_streamtube*)*, 595
go_streamtube_colorbar_tickfont, *608*
go_streamtube_colorbar_tickfont *(*add_streamtube*)*, 595
go_streamtube_colorbar_tickformatstop, *608*
go_streamtube_colorbar_tickformatstop *(*add_streamtube*)*, 595
go_streamtube_colorbar_title, *609*
go_streamtube_colorbar_title *(*add_streamtube*)*, 595
go_streamtube_colorbar_title_font, *611*
go_streamtube_colorbar_title_font *(*add_streamtube*)*, 595
go_streamtube_hoverlabel, *603*
go_streamtube_hoverlabel *(*add_streamtube*)*, 595
go_streamtube_hoverlabel_font, *611*
go_streamtube_hoverlabel_font *(*add_streamtube*)*, 595
go_streamtube_legendgrouptitle, *603*
go_streamtube_legendgrouptitle *(*add_streamtube*)*, 595
go_streamtube_legendgrouptitle_font, *611*
go_streamtube_legendgrouptitle_font *(*add_streamtube*)*, 595
go_streamtube_lighting, *604*
go_streamtube_lighting *(*add_streamtube*)*, 595
go_streamtube_lightposition, *604*
go_streamtube_lightposition *(*add_streamtube*)*, 595
go_streamtube_starts, *604*
go_streamtube_starts *(*add_streamtube*)*, 595
go_sunburst *(*add_sunburst*)*, 613
go_sunburst_domain, *621*
go_sunburst_domain *(*add_sunburst*)*, 613
go_sunburst_hoverlabel, *621*
go_sunburst_hoverlabel *(*add_sunburst*)*, 613
go_sunburst_hoverlabel_font, *625*
go_sunburst_hoverlabel_font *(*add_sunburst*)*, 613
go_sunburst_insidetextfont, *622*
go_sunburst_insidetextfont *(*add_sunburst*)*, 613
go_sunburst_leaf, *622*
go_sunburst_leaf *(*add_sunburst*)*, 613
go_sunburst_legendgrouptitle, *622*
go_sunburst_legendgrouptitle *(*add_sunburst*)*, 613
go_sunburst_legendgrouptitle_font, *625*
go_sunburst_legendgrouptitle_font *(*add_sunburst*)*, 613
go_sunburst_marker, *622*
go_sunburst_marker *(*add_sunburst*)*, 613
go_sunburst_marker_colorbar, *628*
go_sunburst_marker_colorbar *(*add_sunburst*)*, 613
go_sunburst_marker_colorbar_tickfont, *630*
go_sunburst_marker_colorbar_tickfont *(*add_sunburst*)*, 613
go_sunburst_marker_colorbar_tickformatstop, *630*
go_sunburst_marker_colorbar_tickformatstop *(*add_sunburst*)*, 613
go_sunburst_marker_colorbar_title, *631*
go_sunburst_marker_colorbar_title *(*add_sunburst*)*, 613
go_sunburst_marker_colorbar_title_font, *625*
go_sunburst_marker_colorbar_title_font *(*add_sunburst*)*, 613
go_sunburst_marker_line, *628*
go_sunburst_marker_line *(*add_sunburst*)*, 613
go_sunburst_marker_pattern, *628*
go_sunburst_marker_pattern *(*add_sunburst*)*, 613
go_sunburst_outsidetextfont, *623*
go_sunburst_outsidetextfont *(*add_sunburst*)*, 613
go_sunburst_root, *623*
go_sunburst_root *(*add_sunburst*)*, 613
go_sunburst_textfont, *623*
go_sunburst_textfont *(*add_sunburst*)*, 613
go_surface *(*add_surface*)*, 633

go_surface_colorbar, *642*
go_surface_colorbar *(*add_surface*)*, 633
go_surface_colorbar_tickfont, *648*
go_surface_colorbar_tickfont *(*add_surface*)*, 633
go_surface_colorbar_tickformatstop, *648*
go_surface_colorbar_tickformatstop *(*add_surface*)*, 633
go_surface_colorbar_title, *649*
go_surface_colorbar_title *(*add_surface*)*, 633
go_surface_colorbar_title_font, *652*
go_surface_colorbar_title_font *(*add_surface*)*, 633
go_surface_contours, *642*
go_surface_contours *(*add_surface*)*, 633
go_surface_contours_x, *645*
go_surface_contours_x *(*add_surface*)*, 633
go_surface_contours_x_project, *652*
go_surface_contours_x_project *(*add_surface*)*, 633
go_surface_contours_y, *646*
go_surface_contours_y *(*add_surface*)*, 633
go_surface_contours_y_project, *652*
go_surface_contours_y_project *(*add_surface*)*, 633
go_surface_contours_z, *646*
go_surface_contours_z *(*add_surface*)*, 633
go_surface_contours_z_project, *652*
go_surface_contours_z_project *(*add_surface*)*, 633
go_surface_hoverlabel, *642*
go_surface_hoverlabel *(*add_surface*)*, 633
go_surface_hoverlabel_font, *652*
go_surface_hoverlabel_font *(*add_surface*)*, 633
go_surface_legendgrouptitle, *643*
go_surface_legendgrouptitle *(*add_surface*)*, 633
go_surface_legendgrouptitle_font, *652*
go_surface_legendgrouptitle_font *(*add_surface*)*, 633
go_surface_lighting, *644*
go_surface_lighting *(*add_surface*)*, 633
go_surface_lightposition, *644*
go_surface_lightposition *(*add_surface*)*, 633
go_table *(*add_table*)*, 654
go_table_cells, *658*
go_table_cells *(*add_table*)*, 654
go_table_cells_fill, *660*
go_table_cells_fill *(*add_table*)*, 654
go_table_cells_font, *660*
go_table_cells_font *(*add_table*)*, 654
go_table_cells_line, *660*
go_table_cells_line *(*add_table*)*, 654
go_table_domain, *659*
go_table_domain *(*add_table*)*, 654
go_table_header, *659*
go_table_header *(*add_table*)*, 654
go_table_header_fill, *660*
go_table_header_fill *(*add_table*)*, 654
go_table_header_font, *660*
go_table_header_font *(*add_table*)*, 654
go_table_header_line, *660*
go_table_header_line *(*add_table*)*, 654
go_table_hoverlabel, *659*
go_table_hoverlabel *(*add_table*)*, 654
go_table_hoverlabel_font, *660*
go_table_hoverlabel_font *(*add_table*)*, 654
go_table_legendgrouptitle, *659*
go_table_legendgrouptitle *(*add_table*)*, 654
go_table_legendgrouptitle_font, *660*
go_table_legendgrouptitle_font *(*add_table*)*, 654
go_treemap *(*add_treemap*)*, 665
go_treemap_domain, *674*
go_treemap_domain *(*add_treemap*)*, 665
go_treemap_hoverlabel, *674*
go_treemap_hoverlabel *(*add_treemap*)*, 665
go_treemap_hoverlabel_font, *678*
go_treemap_hoverlabel_font *(*add_treemap*)*, 665
go_treemap_insidetextfont, *675*
go_treemap_insidetextfont *(*add_treemap*)*, 665
go_treemap_legendgrouptitle, *675*
go_treemap_legendgrouptitle *(*add_treemap*)*, 665
go_treemap_legendgrouptitle_font, *678*
go_treemap_legendgrouptitle_font *(*add_treemap*)*, 665
go_treemap_marker, *675*

go_treemap_marker (add_treemap), 665
go_treemap_marker_colorbar, *681*
go_treemap_marker_colorbar (add_treemap), 665
go_treemap_marker_colorbar_tickfont, *683*
go_treemap_marker_colorbar_tickfont (add_treemap), 665
go_treemap_marker_colorbar_tickformatstop, *683*
go_treemap_marker_colorbar_tickformatstop (add_treemap), 665
go_treemap_marker_colorbar_title, *684*
go_treemap_marker_colorbar_title (add_treemap), 665
go_treemap_marker_colorbar_title_font, *678*
go_treemap_marker_colorbar_title_font (add_treemap), 665
go_treemap_marker_line, *681*
go_treemap_marker_line (add_treemap), 665
go_treemap_marker_pad, *681*
go_treemap_marker_pad (add_treemap), 665
go_treemap_marker_pattern, *682*
go_treemap_marker_pattern (add_treemap), 665
go_treemap_outsidetextfont, *676*
go_treemap_outsidetextfont (add_treemap), 665
go_treemap_pathbar, *676*
go_treemap_pathbar (add_treemap), 665
go_treemap_pathbar_textfont, *676*
go_treemap_pathbar_textfont (add_treemap), 665
go_treemap_root, *676*
go_treemap_root (add_treemap), 665
go_treemap_textfont, *676*
go_treemap_textfont (add_treemap), 665
go_treemap_tiling, *677*
go_treemap_tiling (add_treemap), 665
go_violin (add_violin), 687
go_violin_box, *693*
go_violin_box (add_violin), 687
go_violin_box_line, *694*
go_violin_box_line (add_violin), 687
go_violin_hoverlabel, *693*
go_violin_hoverlabel (add_violin), 687
go_violin_hoverlabel_font, *699*
go_violin_hoverlabel_font (add_violin), 687
go_violin_legendgrouptitle, *694*
go_violin_legendgrouptitle (add_violin), 687
go_violin_legendgrouptitle_font, *699*
go_violin_legendgrouptitle_font (add_violin), 687
go_violin_line, *694*
go_violin_line (add_violin), 687
go_violin_marker, *695*
go_violin_marker (add_violin), 687
go_violin_marker_line, *694*
go_violin_marker_line (add_violin), 687
go_violin_meanline, *695*
go_violin_meanline (add_violin), 687
go_violin_selected, *696*
go_violin_selected (add_violin), 687
go_violin_selected_marker, *695*
go_violin_selected_marker (add_violin), 687
go_violin_unselected, *697*
go_violin_unselected (add_violin), 687
go_violin_unselected_marker, *695*
go_violin_unselected_marker (add_violin), 687
go_volume (add_volume), 703
go_volume_caps, *711*
go_volume_caps (add_volume), 703
go_volume_caps_x, *715*
go_volume_caps_x (add_volume), 703
go_volume_caps_y, *715*
go_volume_caps_y (add_volume), 703
go_volume_caps_z, *716*
go_volume_caps_z (add_volume), 703
go_volume_colorbar, *712*
go_volume_colorbar (add_volume), 703
go_volume_colorbar_tickfont, *719*
go_volume_colorbar_tickfont (add_volume), 703
go_volume_colorbar_tickformatstop, *719*
go_volume_colorbar_tickformatstop (add_volume), 703
go_volume_colorbar_title, *720*
go_volume_colorbar_title (add_volume), 703
go_volume_colorbar_title_font, *722*

go_volume_colorbar_title_font (add_volume), 703
go_volume_contour, *712*
go_volume_contour (add_volume), 703
go_volume_hoverlabel, *712*
go_volume_hoverlabel (add_volume), 703
go_volume_hoverlabel_font, *722*
go_volume_hoverlabel_font (add_volume), 703
go_volume_legendgrouptitle, *713*
go_volume_legendgrouptitle (add_volume), 703
go_volume_legendgrouptitle_font, *722*
go_volume_legendgrouptitle_font (add_volume), 703
go_volume_lighting, *713*
go_volume_lighting (add_volume), 703
go_volume_lightposition, *713*
go_volume_lightposition (add_volume), 703
go_volume_slices, *714*
go_volume_slices (add_volume), 703
go_volume_slices_x, *715*
go_volume_slices_x (add_volume), 703
go_volume_slices_y, *715*
go_volume_slices_y (add_volume), 703
go_volume_slices_z, *716*
go_volume_slices_z (add_volume), 703
go_volume_spaceframe, *714*
go_volume_spaceframe (add_volume), 703
go_volume_surface, *714*
go_volume_surface (add_volume), 703
go_waterfall (add_waterfall), 724
go_waterfall_connector, *731*
go_waterfall_connector (add_waterfall), 724
go_waterfall_connector_line, *736*
go_waterfall_connector_line (add_waterfall), 724
go_waterfall_decreasing, *731*
go_waterfall_decreasing (add_waterfall), 724
go_waterfall_decreasing_marker, *737*
go_waterfall_decreasing_marker (add_waterfall), 724
go_waterfall_decreasing_marker_line, *736*
go_waterfall_decreasing_marker_line (add_waterfall), 724
go_waterfall_hoverlabel, *731*
go_waterfall_hoverlabel (add_waterfall), 724
go_waterfall_hoverlabel_font, *737*
go_waterfall_hoverlabel_font (add_waterfall), 724
go_waterfall_increasing, *732*
go_waterfall_increasing (add_waterfall), 724
go_waterfall_increasing_marker, *737*
go_waterfall_increasing_marker (add_waterfall), 724
go_waterfall_increasing_marker_line, *736*
go_waterfall_increasing_marker_line (add_waterfall), 724
go_waterfall_insidetextfont, *732*
go_waterfall_insidetextfont (add_waterfall), 724
go_waterfall_legendgrouptitle, *732*
go_waterfall_legendgrouptitle (add_waterfall), 724
go_waterfall_legendgrouptitle_font, *737*
go_waterfall_legendgrouptitle_font (add_waterfall), 724
go_waterfall_outsidetextfont, *733*
go_waterfall_outsidetextfont (add_waterfall), 724
go_waterfall_textfont, *734*
go_waterfall_textfont (add_waterfall), 724
go_waterfall_totals, *734*
go_waterfall_totals (add_waterfall), 724
go_waterfall_totals_marker, *737*
go_waterfall_totals_marker (add_waterfall), 724
go_waterfall_totals_marker_line, *736*
go_waterfall_totals_marker_line (add_waterfall), 724

highlight, *22*, *42*, *56*, *78*, *103*, *124*, *147*, *163*, *175*, *191*, *214*, *230*, *242*, *258*, *273*, *288*, *307*, *325*, *356*, *381*, *405*, *430*, *454*, *474*, *498*, *522*, *546*, *581*, *602*, *621*, *641*, *658*, *674*, *693*, *711*, *731*, *770*, *791*, *807*, *818*, *833*, *849*, *866*, *884*, *905*, *921*, *936*, 1097

highlight(), *937*, *1240*
htmlwidgets::onRender(), *1106*
htmlwidgets::saveWidget(), *1102*, *1105*, *1258*, *1259*

install_chrome, 1099
install_chrome(), *742*

jsonlite::read_json(), *1128*, *1131*, *1177*, *1206*

knit_print.go_figure *(*print.go_figure*)*, 1109

last_plot, 1100

make_subplots, *12*, *13*, *28*, *45*, *62*, *84*, *109*, *130*, *151*, *166*, *177*, *196*, *207*, *220*, *234*, *245*, *260*, *277*, *293*, *309*, *332*, *361*, *385*, *409*, *435*, *457*, *479*, *503*, *526*, *550*, *564*, *572*, *584*, *606*, *624*, *625*, *646*, *647*, *660*, *677*, *698*, *716*, *736*, *770*, *771*, *791*, *807*, *808*, *818*, *833*, *850*, *866*, *884*, *905*, *921*, *936*, 1100, *1244*, *1245*, *1248–1251*, *1254–1257*
make_subplots(), *664*, *937*, *1110*, *1239*

partial_bundle, 1102
partial_bundle(), *1106*
plotly_json, 1104
plotly_json(), *1106*
plotly_obj-modify, 1104
plotly_widget, 1105
plotly_widget(), *1109*, *1258*
plotlyjs_version, 1106
plotlyOutput, 1107
plotlyProxy, 1108
plotlyProxy(), *1107*
plotlyProxyInvoke *(*plotlyProxy*)*, 1108
png::readPNG(), *1165*
print.go_figure, 1109
print.plotly_obj, 1110
print_grid, 1110
print_grid(), *1101*, *1102*
px_area, 1111, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_args, *1166*
px_bar, 1115, *1115*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_bar_polar, *1115*, 1119, *1119*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_box, *1115*, *1119*, 1123, *1123*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_choropleth, *1115*, *1119*, *1123*, 1126, *1126*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_choropleth_map, *1115*, *1119*, *1123*, *1126*, 1130, *1130*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_colors, *32*, *56*, *90*, *104*, *125*, *154*, *192*, *215*, *248*, *263*, *296*, *313*, *334*, *366*, *367*, *390*, *414*, *440*, *441*, *461*, *484*, *507*, *531*, *555*, *589*, *602*, *628*, *642*, *681*, *712*, *766*, *786*, *841*, *857*, *874*, *893*, *913*, *927*, *952*, *958*, *1118*, *1122*, *1129*, *1132*, 1133, *1143*, *1146*, *1163*, *1191*, *1192*, *1199*, *1204*, *1208*, *1211*, *1214*, *1218*, *1221*, *1227*, *1231*, *1233*
px_constant, 1134
px_constant(), *1112*, *1113*, *1116–1118*, *1120*, *1121*, *1124*, *1125*, *1127–1129*,

1131, *1132*, *1138*, *1139*, *1142*, *1143*, *1145*, *1146*, *1149*, *1150*, *1153*, *1154*, *1156*, *1159*, *1160*, *1162*, *1163*, *1168*, *1169*, *1172–1174*, *1177*, *1178*, *1181*, *1184*, *1185*, *1187*, *1188*, *1190*, *1192*, *1194*, *1195*, *1197*, *1198*, *1202*, *1203*, *1206–1208*, *1210*, *1211*, *1214*, *1216*, *1217*, *1220*, *1221*, *1223*, *1224*, *1226*, *1227*, *1229*, *1230*, *1233*, *1236*, *1237*
px_data, 1135
px_data_carshare *(*px_data*)*, 1135
px_data_election *(*px_data*)*, 1135
px_data_election_geojson *(*px_data*)*, 1135
px_data_experiment *(*px_data*)*, 1135
px_data_gapminder *(*px_data*)*, 1135
px_data_iris *(*px_data*)*, 1135
px_data_medals_long *(*px_data*)*, 1135
px_data_medals_wide *(*px_data*)*, 1135
px_data_stocks *(*px_data*)*, 1135
px_data_tips *(*px_data*)*, 1135
px_data_wind *(*px_data*)*, 1135
px_defaults, 1136
px_density_contour, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, 1137, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_density_heatmap, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, 1141, *1141*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_density_map, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, 1145, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_ecdf, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, 1147, *1147*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_funnel, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, 1152, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_funnel_area, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, 1155, *1155*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_get_trendline_results, 1157
px_get_trendline_results(), *1140*, *1200*
px_histogram, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, 1158, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_icicle, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, 1162, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_identity *(*px_constant*)*, 1134
px_imshow, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, 1164, *1164*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_line, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, 1167, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_line_3d, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, 1171, *1171*,

1180, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_line_geo, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, 1175, *1175*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_line_map, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, 1180, *1180*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_line_polar, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, 1183, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_line_ternary, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, 1186, *1186*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1231*, *1234*, *1238*
px_parallel_categories, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, 1190, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1232*, *1234*, *1238*
px_parallel_coordinates, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, 1191, *1191*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1232*, *1234*, *1238*
px_pie, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, 1193, *1193*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1232*, *1234*, *1238*
px_range *(*px_constant*)*, 1134
px_range(), *1112*, *1113*, *1116–1118*, *1120*, *1121*, *1124*, *1125*, *1127–1129*, *1131*, *1132*, *1138*, *1139*, *1142*, *1143*, *1145*, *1146*, *1149*, *1150*, *1153*, *1154*, *1156*, *1159*, *1160*, *1162*, *1163*, *1168*, *1169*, *1172–1174*, *1177*, *1178*, *1181*, *1184*, *1185*, *1187*, *1188*, *1190*, *1192*, *1194*, *1195*, *1197*, *1198*, *1202*, *1203*, *1206–1208*, *1210*, *1211*, *1214*, *1216*, *1217*, *1220*, *1221*, *1223*, *1224*, *1226*, *1227*, *1229*, *1230*, *1233*, *1236*, *1237*
px_scatter, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, 1196, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1232*, *1234*, *1238*
px_scatter(), *740*, *1157*, *1258*
px_scatter_3d, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, 1201, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1232*, *1234*, *1238*
px_scatter_geo, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, 1205, *1205*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1232*, *1234*, *1238*
px_scatter_map, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, 1209, *1209*, *1215*, *1219*, *1222*, *1225*, *1228*, *1232*, *1234*, *1238*
px_scatter_matrix, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*,

1175, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, 1213, *1219*, *1222*, *1225*, *1228*, *1232*, *1234*, *1238*
px_scatter_polar, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, 1215, *1215*, *1222*, *1225*, *1228*, *1232*, *1234*, *1238*
px_scatter_ternary, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, 1219, *1219*, *1225*, *1228*, *1232*, *1234*, *1238*
px_strip, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, 1222, *1222*, *1228*, *1232*, *1234*, *1238*
px_sunburst, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, 1225, *1225*, *1232*, *1234*, *1238*
px_timeline, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, 1228, *1228*, *1234*, *1238*
px_treemap, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, 1232, *1232*, *1238*
px_violin, *1115*, *1119*, *1123*, *1126*, *1130*, *1133*, *1141*, *1144*, *1147*, *1151*, *1155*, *1157*, *1161*, *1164*, *1166*, *1171*, *1175*, *1180*, *1182*, *1186*, *1189*, *1191*, *1193*, *1195*, *1200*, *1205*, *1209*, *1212*, *1215*, *1219*, *1222*, *1225*, *1228*, *1232*, *1234*, 1235

register_template *(*get_template*)*, 745
remove_chrome *(*install_chrome*)*, 1099
remove_chrome(), *742*
renderPlotly *(*plotlyOutput*)*, 1107
renderPlotly(), *1108*
reset_px_defaults *(*px_defaults*)*, 1136

save_image *(*chrome_exporter*)*, 741
save_image(), *1099*, *1109*, *1241*, *1259*
schema, 1238
select_annotations *(*update_annotations*)*, 1242
select_coloraxes *(*update_coloraxes*)*, 1243
select_geos *(*update_geos*)*, 1244
select_layout_images *(*update_layout_images*)*, 1246
select_legends *(*update_legends*)*, 1247
select_maps *(*update_maps*)*, 1248
select_polars *(*update_polars*)*, 1249
select_scenes *(*update_scenes*)*, 1250
select_selections *(*update_selections*)*, 1251
select_shapes *(*update_shapes*)*, 1252
select_smiths *(*update_smiths*)*, 1254
select_ternaries *(*update_ternaries*)*, 1255
select_traces, 1239
select_xaxes *(*update_xaxes*)*, 1256
select_yaxes *(*update_yaxes*)*, 1257
set_crosstalk_keys, 1240
set_crosstalk_keys(), *1097*, *1099*
set_last_plot *(*last_plot*)*, 1100
shiny::getCurrentOutputInfo(), *1107*
stats::lm(), *1157*

template_names *(*get_template*)*, 745
template_names(), *1114*, *1119*, *1122*, *1126*, *1130*, *1133*, *1140*, *1144*, *1147*, *1151*, *1154*, *1156*, *1161*, *1164*, *1171*, *1175*, *1179*, *1182*, *1186*, *1189*, *1191*, *1192*, *1195*, *1200*, *1204*, *1209*, *1212*, *1215*,

1218, *1222*, *1225*, *1227*, *1231*, *1234*, *1237*
TeX, 1241
TeX(), *741*, *743*

update_annotations, 1242
update_coloraxes, 1243
update_geos, 1244
update_layout, *947*, *1242–1244*, 1245, *1246–1252*, *1254–1257*
update_layout(), *937*, *1239*
update_layout_images, 1246
update_legends, 1247
update_maps, 1248
update_polars, 1249
update_scenes, 1250
update_selections, 1251
update_shapes, 1252
update_smiths, 1254
update_ternaries, 1255
update_traces *(*select_traces*)*, 1239
update_traces(), *937*
update_xaxes, 1256
update_xaxes(), *1100*
update_yaxes, 1257

write_html, 1258