Package 'plotly'

September 26, 2016

Title Create Interactive Web Graphics via 'plotly.js'

Version 4.5.2

License MIT + file LICENSE

Description Easily translate 'ggplot2' graphs to an interactive web-based version and/or create custom web-based visualizations directly from R. Once uploaded to a 'plotly' account, 'plotly' graphs (and the data behind them) can be viewed and modified in a web browser.

URL https://plot.ly/r, https://cpsievert.github.io/plotly_book/, https://github.com/ropensci/plotly

BugReports https://github.com/ropensci/plotly/issues

Depends R (>= 3.2.0), ggplot2 (>= 2.1.0)

Imports scales, httr, jsonlite, magrittr, digest, viridisLite, base64enc, htmlwidgets, tidyr, dplyr, tibble, hexbin, lazyeval (>= 0.2.0), purrr

Suggests MASS, maps, ggthemes, GGally, testthat, knitr, devtools, shiny (>= 0.14), htmltools, curl, rmarkdown, RColorBrewer, Rserve, RSclient, broom, webshot, listviewer

LazyData true

RoxygenNote 5.0.1

NeedsCompilation no

Author Carson Sievert [aut, cre], Chris Parmer [aut, cph], Toby Hocking [aut], Scott Chamberlain [aut], Karthik Ram [aut], Marianne Corvellec [aut], Pedro Despouy [aut]

Maintainer Carson Sievert <cpsievert1@gmail.com>

Repository CRAN

Date/Publication 2016-09-26 17:01:59

R topics documented:

`add_annotations` *Add an annotation(s) to a plot*

Description

Add an annotation(s) to a plot

Usage

```
add_annotations(p, text = NULL, ..., data = NULL, inherit = TRUE)
```

Arguments

`p`	a plotly object
`text`	annotation text (required).
`...`	these arguments are documented at https://plot.ly/r/reference/#layout-annotations
`data`	a data frame.
`inherit`	inherit attributes from `plot_ly()`?

Author(s)

Carson Sievert

Examples

```
# single annotation
plot_ly(mtcars, x = ~wt, y = ~mpg) %>%
  slice(which.max(mpg)) %>%
  add_annotations(text = "Good mileage")

# multiple annotations
plot_ly(mtcars, x = ~wt, y = ~mpg) %>%
  filter(gear == 5) %>%
  add_annotations("five cylinder", ax = 40)
```

add_data *Add data to a plotly visualization*

Description

Add data to a plotly visualization

Usage

```
add_data(p, data = NULL)
```

Arguments

p	a plotly visualization
data	a data frame.

Examples

```
plot_ly() %>% add_data(economics) %>% add_trace(x = ~date, y = ~pce)
```

add_fun *Apply function to plot, without modifying data*

Description

Useful when you need two or more layers that apply a summary statistic to the original data.

Usage

```
add_fun(p, fun, ...)
```

Arguments

p	a plotly object.
fun	a function. Should take a plotly object as input and return a modified plotly object.
...	arguments passed to fun.

Examples

```
txhousing %>%
  group_by(city) %>%
  plot_ly(x = ~date, y = ~median) %>%
  add_lines(alpha = 0.2, name = "Texan Cities") %>%
  add_fun(function(plot) {
    plot %>% filter(city == "Houston") %>% add_lines(name = "Houston")
  }) %>%
  add_fun(function(plot) {
    plot %>% filter(city == "San Antonio") %>% add_lines(name = "San Antonio")
  })

plot_ly(mtcars, x = ~wt, y = ~mpg) %>%
  add_markers() %>%
  add_fun(function(p) {
    p %>% slice(which.max(mpg)) %>%
      add_annotations("Good mileage")
  }) %>%
  add_fun(function(p) {
    p %>% slice(which.min(mpg)) %>%
      add_annotations(text = "Bad mileage")
  })
```

---

add_trace *Add trace(s) to a plotly visualization*

Description

Add trace(s) to a plotly visualization

Usage

```
add_trace(p, ..., data = NULL, inherit = TRUE)

add_markers(p, x = NULL, y = NULL, z = NULL, ..., data = NULL,
  inherit = TRUE)

add_text(p, x = NULL, y = NULL, z = NULL, text = NULL, ...,
  data = NULL, inherit = TRUE)

add_paths(p, x = NULL, y = NULL, z = NULL, ..., data = NULL,
  inherit = TRUE)

add_lines(p, x = NULL, y = NULL, z = NULL, ..., data = NULL,
  inherit = TRUE)
```

```
add_segments(p, x = NULL, y = NULL, xend = NULL, yend = NULL, ...,
  data = NULL, inherit = TRUE)

add_polygons(p, x = NULL, y = NULL, ..., data = NULL, inherit = TRUE)

add_ribbons(p, x = NULL, ymin = NULL, ymax = NULL, ..., data = NULL,
  inherit = TRUE)

add_area(p, r = NULL, t = NULL, ..., data = NULL, inherit = TRUE)

add_pie(p, values = NULL, labels = NULL, ..., data = NULL,
  inherit = TRUE)

add_bars(p, x = NULL, y = NULL, ..., data = NULL, inherit = TRUE)

add_histogram(p, x = NULL, y = NULL, ..., data = NULL, inherit = TRUE)

add_histogram2d(p, x = NULL, y = NULL, z = NULL, ..., data = NULL,
  inherit = TRUE)

add_histogram2dcontour(p, x = NULL, y = NULL, z = NULL, ...,
  data = NULL, inherit = TRUE)

add_heatmap(p, x = NULL, y = NULL, z = NULL, ..., data = NULL,
  inherit = TRUE)

add_contour(p, z = NULL, ..., data = NULL, inherit = TRUE)

add_boxplot(p, x = NULL, y = NULL, ..., data = NULL, inherit = TRUE)

add_surface(p, z = NULL, ..., data = NULL, inherit = TRUE)

add_mesh(p, x = NULL, y = NULL, z = NULL, ..., data = NULL,
  inherit = TRUE)

add_scattergeo(p, ...)

add_choropleth(p, z = NULL, ..., data = NULL, inherit = TRUE)
```

Arguments

p	a plotly object
...	These arguments are documented at https://plot.ly/r/reference/ Note that acceptable arguments depend on the value of type.
data	A data frame (optional).
inherit	inherit attributes from plot_ly()?
x	the x variable.

y	the y variable.
z	a numeric matrix
text	textual labels.
xend	"final" x position (in this context, x represents "start")
yend	"final" y position (in this context, y represents "start")
ymin	a variable used to define the lower boundary of a polygon.
ymax	a variable used to define the upper boundary of a polygon.
r	For polar chart only. Sets the radial coordinates.
t	For polar chart only. Sets the radial coordinates.
values	the value to associated with each slice of the pie.
labels	the labels (categories) corresponding to values.

Author(s)

Carson Sievert

References

https://plot.ly/r/reference/

See Also

plot_ly()

Examples

```
p <- plot_ly(economics, x = ~date, y = ~uempmed)
p
p %>% add_markers()
p %>% add_lines()
p %>% add_text(text = ".")

# attributes declared in plot_ly() carry over to downstream traces,
# but can be overwritten
plot_ly(economics, x = ~date, y = ~uempmed, color = I("red")) %>%
  add_lines() %>%
  add_markers(color = ~pop) %>%
  layout(showlegend = FALSE)

txhousing %>%
  group_by(city) %>%
  plot_ly(x = ~date, y = ~median) %>%
  add_lines(fill = "black")

ggplot2::map_data("world", "canada") %>%
  group_by(group) %>%
  plot_ly(x = ~long, y = ~lat) %>%
```

```
  add_polygons(hoverinfo = "none") %>%
  add_markers(text = ~paste(name, "<br />", pop), hoverinfo = "text",
    data = maps::canada.cities) %>%
  layout(showlegend = FALSE)

plot_ly(economics, x = ~date) %>%
  add_ribbons(ymin = ~pce - 1e3, ymax = ~pce + 1e3)
p <- plot_ly(plotly::wind, r = ~r, t = ~t) %>% add_area(color = ~nms)
layout(p, radialaxis = list(ticksuffix = "%"), orientation = 270)
ds <- data.frame(
  labels = c("A", "B", "C"),
  values = c(10, 40, 60)
)

plot_ly(ds, labels = ~labels, values = ~values) %>%
  add_pie() %>%
  layout(title = "Basic Pie Chart using Plotly")
library(dplyr)
mtcars %>%
  count(vs) %>%
  plot_ly(x = ~vs, y = ~n) %>%
  add_bars()

plot_ly(x = ~rnorm(100)) %>% add_histogram()
plot_ly(x = ~LETTERS, y = ~LETTERS) %>% add_histogram2d()
z <- as.matrix(table(LETTERS, LETTERS))
plot_ly(x = ~LETTERS, y = ~LETTERS, z = ~z) %>% add_histogram2d()
plot_ly(MASS::geyser, x = ~waiting, y = ~duration) %>%
add_histogram2dcontour()
plot_ly(z = ~volcano) %>% add_heatmap()
plot_ly(z = ~volcano) %>% add_contour()
plot_ly(mtcars, x = ~factor(vs), y = ~mpg) %>% add_boxplot()
plot_ly(z = ~volcano) %>% add_surface()
plot_ly(x = c(0, 0, 1), y = c(0, 1, 0), z = c(0, 0, 0)) %>% add_mesh()
```

`as.widget` *Convert a plotly object to an htmlwidget object*

Description

This function was deprecated in 4.0.0, as plotly objects are now htmlwidget objects, so there is no need to convert them.

Usage

```
as.widget(x, ...)
```

Arguments

x	a plotly object.
...	other options passed onto htmlwidgets::createWidget

---

as_widget *Convert a list to a plotly htmlwidget object*

Description

Convert a list to a plotly htmlwidget object

Usage

```
as_widget(x, ...)
```

Arguments

x	a plotly object.
...	other options passed onto htmlwidgets::createWidget

Examples

```
trace <- list(x = 1, y = 1)
obj <- list(data = list(trace), layout = list(title = "my plot"))
as_widget(obj)
```

---

bbox *Estimate bounding box of a rotated string*

Description

Estimate bounding box of a rotated string

Usage

```
bbox(txt = "foo", angle = 0, size = 12)
```

Arguments

txt	a character string of length 1
angle	sets the angle of the tick labels with respect to the horizontal (e.g., 'tickangle' of -90 draws the tick labels vertically)
size	vertical size of a character

References

https://www.dropbox.com/s/nc6968prgw8ne4w/bbox.pdf?dl=0

`colorbar` *Modify the colorbar*

Description

Modify the colorbar

Usage

```
colorbar(p, ...)
```

Arguments

`p`	a plotly object
`...`	arguments are documented here https://plot.ly/r/reference/#scatter-marker-colorbar.

Author(s)

Carson Sievert

Examples

```
plot_ly(mpg, x = ~cty, y = ~hwy, color = ~cyl) %>%
  colorbar(len = 0.5)
```

`config` *Set the default configuration for plotly*

Description

Set the default configuration for plotly

Usage

```
config(p, ...)
```

Arguments

`p`	a plotly object
`...`	these arguments are documented at https://github.com/plotly/plotly.js/blob/master/src/plot_api/plot_config.js

Author(s)

Carson Sievert

Examples

```
## Not run:
config(plot_ly(), displaylogo = FALSE, modeBarButtonsToRemove = list('sendDataToCloud'))

## End(Not run)
```

---

embed_notebook *Embed a plotly figure as an iframe into a Jupyter Notebook*

Description

Embed a plotly figure as an iframe into a Jupyter Notebook

Usage

```
embed_notebook(x, width = NULL, height = NULL,
  file = paste0("plotlyJupyterHTML/", digest::digest(Sys.time()), ".html"))
```

Arguments

x	a plotly object
width	attribute of the iframe. If NULL, the width in plot_ly is used. If that is also NULL, '100%' is the default.
height	attribute of the iframe. If NULL, the height in plot_ly is used. If that is also NULL, '400px' is the default.
file	a filename for saving the standalone HTML (only used if x is a non-figure object)

---

event_data *Access plotly user input event data in shiny*

Description

This function must be called within a reactive shiny context.

Usage

```
event_data(event = c("plotly_hover", "plotly_click", "plotly_selected",
  "plotly_relayout"), source = "A",
  session = shiny::getDefaultReactiveDomain())
```

Arguments

event	The type of plotly event. Currently 'plotly_hover', 'plotly_click', 'plotly_selected', and 'plotly_relayout' are supported.
source	Which plot should the listener be tied to? This (character string) should match the value of source in plot_ly().
session	a shiny session object (the default should almost always be used).

Author(s)

Carson Sievert

Examples

```
## Not run:
shiny::runApp(system.file("examples", "plotlyEvents", package = "plotly"))

## End(Not run)
```

---

export *Export a plotly graph to a static file*

Description

Export a plotly graph to a static file

Usage

```
export(p, file = "plotly.png", ...)
```

Arguments

p	a plotly or ggplot object.
file	a filename. See the file argument of webshot::webshot for valid extensions.
...	arguments passed onto webshot::webshot

Examples

```
## Not run:
export(plot_ly(economics, x = ~date, y = ~pce))

## End(Not run)
```

geom2trace *Convert a "basic" geoms to a plotly.js trace.*

Description

This function makes it possible to convert ggplot2 geoms that are not included with ggplot2 itself. Users shouldn't need to use this function. It exists purely to allow other package authors to write their own conversion method(s).

Usage

```
geom2trace(data, params, p)
```

Arguments

data	the data returned by `plotly::to_basic`.
params	parameters for the geom, statistic, and 'constant' aesthetics
p	a ggplot2 object (the conversion may depend on scales, for instance).

get_figure *Request a figure object*

Description

Figure objects work in the same way as plotly objects, but when printed, they overwrite the already existing plotly object (instead of creating a new plotly).

Usage

```
get_figure(username, id)
```

Arguments

username	corresponding username for the figure.
id	of the Plotly figure.

References

https://plot.ly/rest/

Examples

```
## Not run:
 # Anyone can obtain the information for a particular plot
 fig <- get_figure("cpsievert", "559")
 # If you have proper credentials, you can easily modify
 layout(fig, title = paste("Modified on ", Sys.time()))

## End(Not run)
```

`gg2list` *Convert a ggplot to a list.*

Description

Convert a ggplot to a list.

Usage

```
gg2list(p, width = NULL, height = NULL, tooltip = "all", layerData = 1,
  originalData = TRUE, source = "A", ...)
```

Arguments

`p`	ggplot2 plot.
`width`	Width of the plot in pixels (optional, defaults to automatic sizing).
`height`	Height of the plot in pixels (optional, defaults to automatic sizing).
`tooltip`	a character vector specifying which aesthetic tooltips to show in the tooltip. The default, "all", means show all the aesthetic tooltips (including the unofficial "text" aesthetic).
`layerData`	data from which layer should be returned?
`originalData`	should the "original" or "scaled" data be returned?
`source`	Only relevant for event_data.
`...`	currently not used

Value

a 'built' plotly object (list with names "data" and "layout").

ggplotly *Create plotly graphs using ggplot2 syntax*

Description

See up-to-date documentation and examples at https://plot.ly/ggplot2

Usage

```
ggplotly(p = ggplot2::last_plot(), width = NULL, height = NULL,
  tooltip = "all", layerData = 1, originalData = TRUE, source = "A",
  ...)
```

Arguments

p	a ggplot object.
width	Width of the plot in pixels (optional, defaults to automatic sizing).
height	Height of the plot in pixels (optional, defaults to automatic sizing).
tooltip	a character vector specifying which aesthetic mappings to show in the tooltip. The default, "all", means show all the aesthetic mappings (including the unofficial "text" aesthetic). The order of variables here will also control the order they appear. For example, use `tooltip = c("y", "x", "colour")` if you want y first, x second, and colour last.
layerData	data from which layer should be returned?
originalData	should the "original" or "scaled" data be returned?
source	Only relevant for event_data.
...	arguments passed onto methods.

Value

a plotly object

Author(s)

Carson Sievert

See Also

signup(), plot_ly()

Examples

```
## Not run:
# simple example
ggiris <- qplot(Petal.Width, Sepal.Length, data = iris, color = Species)
ggplotly(ggiris)

data(canada.cities, package = "maps")
viz <- ggplot(canada.cities, aes(long, lat)) +
  borders(regions = "canada") +
  coord_equal() +
  geom_point(aes(text = name, size = pop), colour = "red", alpha = 1/2)
ggplotly(viz, tooltip = c("text", "size"))

## End(Not run)
```

`hide_colorbar` *Hide color bar(s)*

Description

Hide color bar(s)

Usage

```
hide_colorbar(p)
```

Arguments

p	a plotly object.

See Also

`hide_legend()`

Examples

```
plot_ly(economics, x = ~date, y = ~unemploy / pop, color = ~pop) %>%
  add_markers() %>%
  hide_colorbar()
```

hide_guides *Hide guides (legends and colorbars)*

Description

Hide guides (legends and colorbars)

Usage

```
hide_guides(p)
```

Arguments

p	a plotly object.

See Also

hide_legend(), hide_colorbar()

hide_legend *Hide legend*

Description

Hide legend

Usage

```
hide_legend(p)
```

Arguments

p	a plotly object.

See Also

hide_legend()

Examples

```
plot_ly(economics, x = ~date, y = ~unemploy / pop, color = ~pop) %>%
  add_markers() %>%
  hide_colorbar()
```

hobbs *Hobbs data*

Description

Description TBD.

Usage

```
hobbs
```

Format

A data frame with three variables: `r`, `t`, `nms`.

knit_print.plotly_figure *Embed a plotly figure as an iframe in a knitr doc*

Description

Embed a plotly figure as an iframe in a knitr doc

Usage

```
knit_print.plotly_figure(x, options, ...)
```

Arguments

`x`	a plotly figure object
`options`	knitr options.
`...`	placeholder.

References

https://github.com/yihui/knitr/blob/master/vignettes/knit_print.Rmd

last_plot *Retrieve the last plot to be modified or created.*

Description

Retrieve the last plot to be modified or created.

Usage

```
last_plot()
```

See Also

last_plot

layout *Modify the layout of a plotly visualization*

Description

Modify the layout of a plotly visualization

Usage

```
layout(p, ..., data = NULL)
```

Arguments

p	A plotly object.
...	Arguments to the layout object. For documentation, see https://plot.ly/r/reference/#Layout_and_layout_style_objects
data	A data frame to associate with this layout (optional). If not provided, arguments are evaluated using the data frame in plot_ly().

Author(s)

Carson Sievert

mic *Mic data*

Description

Description TBD.

Usage

```
mic
```

Format

A data frame with three variables: `r`, `t`, `nms`.

offline *Plotly Offline*

Description

Deprecated in version 2.0 (offline plots are now the default)

Usage

```
offline(p, height, width, out_dir, open_browser)
```

Arguments

`p`	a plotly object
`height`	A valid CSS unit. (like "100%", "600px", "auto") or a number, which will be coerced to a string and have "px" appended.
`width`	A valid CSS unit. (like "100%", "600px", "auto") or a number, which will be coerced to a string and have "px" appended.
`out_dir`	a directory to place the visualization. If `NULL`, a temporary directory is used when the offline object is printed.
`open_browser`	open the visualization after creating it?

Value

a plotly object of class "offline"

Author(s)

Carson Sievert

plotly *Main interface to plotly*

Description

Deprecated: see signup for credentials/configuration storage details. See ggplotly for the new ggplot2 interface.

Usage

```
plotly(username, key)
```

Arguments

username	plotly username
key	plotly API key

plotly-shiny *Shiny bindings for plotly*

Description

Output and render functions for using plotly within Shiny applications and interactive Rmd documents.

Usage

```
plotlyOutput(outputId, width = "100%", height = "400px")

renderPlotly(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like "100%", "400px", "auto") or a number, which will be coerced to a string and have "px" appended.
expr	An expression that generates a plotly
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

plotly_build *'Build' (i.e., evaluate) a plotly object*

Description

This generic function creates the list object sent to plotly.js for rendering. Using this function can be useful for overriding defaults provided by ggplotly/plot_ly or for debugging rendering errors.

Usage

```
plotly_build(p)
```

Arguments

p	a ggplot object, or a plotly object, or a list.

Examples

```
p <- plot_ly(economics, x = ~date, y = ~pce)
# the unevaluated plotly object
str(p)
# the evaluated data
str(plotly_build(p)$x$data)
```

plotly_data *Obtain data associated with a plotly graph*

Description

plotly_data() returns data associated with a plotly visualization (if there are multiple data frames, by default, it returns the most recent one).

Usage

```
plotly_data(p, id = p$x$cur_data)

## S3 method for class 'plotly'
groups(x)

## S3 method for class 'plotly'
ungroup(x, ...)

## S3 method for class 'plotly'
group_by_(.data, ..., .dots, add = FALSE)
```

```
## S3 method for class 'plotly'
summarise_(.data, ..., .dots)

## S3 method for class 'plotly'
mutate_(.data, ..., .dots)

## S3 method for class 'plotly'
arrange_(.data, ..., .dots)

## S3 method for class 'plotly'
select_(.data, ..., .dots)

## S3 method for class 'plotly'
filter_(.data, ..., .dots)

## S3 method for class 'plotly'
distinct_(.data, ..., .dots)

## S3 method for class 'plotly'
slice_(.data, ..., .dots)

## S3 method for class 'plotly'
rename_(.data, ..., .dots)

## S3 method for class 'plotly'
transmute_(.data, ..., .dots)
```

Arguments

p	a plotly visualization
id	a character string or number referencing an "attribute layer".
x	a plotly visualization
...	stuff passed onto the relevant method
.data	a plotly visualization
.dots	Used to work around non-standard evaluation. See vignette("nse") for details
add	By default, when add = FALSE, group_by will override existing groups. To instead add to the existing groups, use add = TRUE

Examples

```
# use group_by() to define groups of visual markings
p <- txhousing %>%
  group_by(city) %>%
  plot_ly(x = ~date, y = ~sales)
p
```

```
# plotly objects preserve data groupings
groups(p)
plotly_data(p)

# dplyr verbs operate on plotly objects as if they were data frames
p <- economics %>%
  plot_ly(x = ~date, y = ~unemploy / pop) %>%
  add_lines() %>%
  mutate(rate = unemploy / pop) %>%
  filter(rate == max(rate))
plotly_data(p)
add_markers(p)
layout(p, annotations = list(x = ~date, y = ~rate, text = "peak"))

# use group_by() + do() + subplot() for trellis displays
d <- group_by(mpg, drv)
plots <- do(d, p = plot_ly(., x = ~cty, name = ~drv))
subplot(plots[["p"]], nrows = 3, shareX = TRUE)

# arrange displays by their mean
means <- summarise(d, mn = mean(cty, na.rm = TRUE))
means %>%
  dplyr::left_join(plots) %>%
  arrange(mn) %>%
  subplot(nrows = NROW(.), shareX = TRUE)

# more dplyr verbs applied to plotly objects
p <- mtcars %>%
  plot_ly(x = ~wt, y = ~mpg, name = "scatter trace") %>%
  add_markers()
p %>% slice(1) %>% plotly_data()
p %>% slice(1) %>% add_markers(name = "first observation")
p %>% filter(cyl == 4) %>% plotly_data()
p %>% filter(cyl == 4) %>% add_markers(name = "four cylinders")
```

`plotly_empty` *Create a complete empty plotly graph.*

Description

Useful when used with `subplot()`

Usage

```
plotly_empty(...)
```

Arguments

`...`	arguments passed onto `plot_ly()`

plotly_IMAGE *Create/Modify plotly images*

Description

The images endpoint turn a plot (which may be given in multiple forms) into an image of the desired format.

Usage

```
plotly_IMAGE(x, width = 1000, height = 500, format = "png", scale = 1,
  out_file, ...)
```

Arguments

x	either a plotly object or a list.
width	Image width in pixels
height	Image height in pixels
format	The desired image format 'png', 'jpeg', 'svg', 'pdf', 'eps', or 'webp'
scale	Both png and jpeg formats will be scaled beyond the specified width and height by this number.
out_file	A filename for writing the image to a file.
...	arguments passed onto `httr::POST`

Examples

```
## Not run:
p <- plot_ly(x = 1:10)
Png <- plotly_IMAGE(p, out_file = "plotly-test-image.png")
Jpeg <- plotly_IMAGE(p, format = "jpeg", out_file = "plotly-test-image.jpeg")
Svg <- plotly_IMAGE(p, format = "svg",  out_file = "plotly-test-image.svg")
Pdf <- plotly_IMAGE(p, format = "pdf",  out_file = "plotly-test-image.pdf")

## End(Not run)
```

plotly_json *Inspect JSON sent to plotly.js*

Description

This function is useful for obtaining/viewing/debugging JSON sent to plotly.js.

Usage

```
plotly_json(p = last_plot(), jsonedit = interactive(), ...)
```

Arguments

p	a plotly or ggplot object.
jsonedit	use `listviewer::jsonedit` to view the JSON?
...	other options passed onto `listviewer::jsonedit`

Examples

```
plotly_json(plot_ly())
plotly_json(plot_ly(), FALSE)
```

---

plotly_POST *Create/Modify plotly graphs*

Description

Create and modify graphs on your plotly account via plotly's REST API https://plot.ly/rest/

Usage

```
plotly_POST(x = last_plot(), filename = NULL, fileopt = "overwrite",
  sharing = c("public", "private", "secret"))
```

Arguments

x	either a ggplot object, a plotly object, or a list.
filename	character string describing the name of the plot in your plotly account. Use / to specify directories. If a directory path does not exist it will be created. If this argument is not specified and the title of the plot exists, that will be used for the filename.
fileopt	character string describing whether to create a "new" plotly, "overwrite" an existing plotly, "append" data to existing plotly, or "extend" it.
sharing	If 'public', anyone can view this graph. It will appear in your profile and can appear in search engines. You do not need to be logged in to Plotly to view this chart. If 'private', only you can view this plot. It will not appear in the Plotly feed, your profile, or search engines. You must be logged in to Plotly to view this graph. You can privately share this graph with other Plotly users in your online Plotly account and they will need to be logged in to view this plot. If 'secret', anyone with this secret link can view this chart. It will not appear in the Plotly feed, your profile, or search engines. If it is embedded inside a webpage or an IPython notebook, anybody who is viewing that page will be able to view the graph. You do not need to be logged in to view this plot.

Value

An R object created by mapping the JSON content of the plotly API response to its R equivalent.

Author(s)

Carson Sievert

See Also

plot_ly(), signup()

Examples

```
## Not run:
p <- plot_ly(mtcars, x = ~factor(vs))
plotly_POST(p, filename = "mtcars-bar-plot")

## End(Not run)
```

---

plot_geo *Initiate a plotly-geo object*

Description

Use this function instead of plot_ly() to initialize a plotly-geo object. This enforces the entire plot so use the scattergeo trace type, and enables higher level geometries like add_polygons() to work

Usage

```
plot_geo(data = data.frame(), ...)
```

Arguments

data	A data frame (optional).
...	arguments passed along to plot_ly().

Author(s)

Carson Sievert

See Also

plot_ly(), plot_mapbox(), ggplotly()

Examples

```
map_data("world", "canada") %>%
  group_by(group) %>%
  plot_geo(x = ~long, y = ~lat) %>%
  add_markers(size = I(1))
```

`plot_ly` *Initiate a plotly visualization*

Description

Transform data into a plotly visualization.

Usage

```
plot_ly(data = data.frame(), ..., type = NULL, color, colors = NULL,
  alpha = 1, symbol, symbols = NULL, size, sizes = c(10, 100), linetype,
  linetypes = NULL, split, width = NULL, height = NULL, source = "A")
```

Arguments

`data`	A data frame (optional).
`...`	These arguments are documented at https://plot.ly/r/reference/ Note that acceptable arguments depend on the value of `type`.
`type`	A character string describing the type of trace.
`color`	A formula containing a name or expression. Values are scaled and mapped to color codes based on the value of `colors` and `alpha`. To avoid scaling, wrap with `I()`, and provide value(s) that can be converted to rgb color codes by `col2rgb()`.
`colors`	Either a colorbrewer2.org palette name (e.g. "YlOrRd" or "Blues"), or a vector of colors to interpolate in hexadecimal "#RRGGBB" format, or a color interpolation function like `colorRamp()`.
`alpha`	A number between 0 and 1 specifying the alpha channel applied to color.
`symbol`	A formula containing a name or expression. Values are scaled and mapped to symbols based on the value of `symbols`. To avoid scaling, wrap with `I()`, and provide valid `pch()` values and/or valid plotly symbol(s) as a string
`symbols`	A character vector of symbol types. Either valid pch or plotly symbol codes may be supplied.
`size`	A formula containing a name or expression yielding a numeric vector. Values are scaled according to the range specified in `sizes`.
`sizes`	A numeric vector of length 2 used to scale sizes to pixels.

linetype	A formula containing a name or expression. Values are scaled and mapped to linetypes based on the value of linetypes. To avoid scaling, wrap with I().
linetypes	A character vector of line types. Either valid par (lty) or plotly dash codes may be supplied.
split	A formula containing a name or expression. Similar to group_by(), but ensures at least one trace for each unique value. This replaces the functionality of the (now deprecated) group argument.
width	Width in pixels (optional, defaults to automatic sizing).
height	Height in pixels (optional, defaults to automatic sizing).
source	Only relevant for event_data.

Details

There are a number of "visual properties" that aren't included in the officical Reference section (see below).

Author(s)

Carson Sievert

See Also

- For initializing a plotly-geo object: plot_geo().
- For initializing a plotly-mapbox object: plot_mapbox().
- For translating a ggplot2 object to a plotly object: ggplotly().
- For modifying any plotly object: layout(), add_trace(), style()
-

Examples

```
## Not run:

# plot_ly() tries to create a sensible plot based on the information you
# give it. If you don't provide a trace type, plot_ly() will infer one.
plot_ly(economics, x = ~pop)
plot_ly(economics, x = ~date, y = ~pop)
# plot_ly() doesn't require data frame(s), which allows one to take
# advantage of trace type(s) designed specifically for numeric matrices
plot_ly(z = ~volcano)
plot_ly(z = ~volcano, type = "surface")

# plotly has a functional interface: every plotly function takes a plotly
# object as it's first input argument and returns a modified plotly object
add_lines(plot_ly(economics, x = ~date, y = ~unemploy/pop))

# To make code more readable, plotly imports the pipe operator from magrittr
economics %>% plot_ly(x = ~date, y = ~unemploy/pop) %>% add_lines()
```

```
# Attributes defined via plot_ly() set 'global' attributes that
# are carried onto subsequent traces, but those may be over-written
plot_ly(economics, x = ~date, color = I("black")) %>%
 add_lines(y = ~uempmed) %>%
 add_lines(y = ~psavert, color = I("red"))

# Attributes are documented in the figure reference -> https://plot.ly/r/reference
# You might notice plot_ly() has named arguments that aren't in this figure
# reference. These arguments make it easier to map abstract data values to
# visual attributes.
p <- plot_ly(iris, x = ~Sepal.Width, y = ~Sepal.Length)
add_markers(p, color = ~Petal.Length, size = ~Petal.Length)
add_markers(p, color = ~Species)
add_markers(p, color = ~Species, colors = "Set1")
add_markers(p, symbol = ~Species)
add_paths(p, linetype = ~Species)


## End(Not run)
```

`plot_mapbox` *Initiate a plotly-mapbox object*

Description

Use this function instead of `plot_ly()` to initialize a plotly-mapbox object. This enforces the entire plot so use the scattermapbox trace type, and enables higher level geometries like `add_polygons()` to work

Usage

```
plot_mapbox(data = data.frame(), ...)
```

Arguments

`data`	A data frame (optional).
`...`	arguments passed along to `plot_ly()`. They should be valid scattermapbox attributes - https://plot.ly/r/reference/#scattermapbox. Note that x/y can also be used in place of lat/lon.

Author(s)

Carson Sievert

See Also

`plot_ly()`, `plot_geo()`, `ggplotly()`

Examples

```
## Not run:

map_data("world", "canada") %>%
  group_by(group) %>%
  plot_mapbox(x = ~long, y = ~lat) %>%
  add_polygons() %>%
  layout(
    mapbox = list(
      center = list(lat = ~median(lat), lon = ~median(long))
    )
  )

## End(Not run)
```

---

print.plotly_figure *Print a plotly figure object*

Description

Print a plotly figure object

Usage

```
## S3 method for class 'plotly_figure'
print(x, ...)
```

Arguments

x	a plotly figure object
...	additional arguments (currently ignored)

---

rangeslider *Add a range slider to the x-axis*

Description

Add a range slider to the x-axis

Usage

```
rangeslider(p, ...)
```

Arguments

p	plotly object.
...	these arguments are documented here https://plot.ly/r/reference/#layout-xaxis-rangeslider

Author(s)

Carson Sievert

Examples

```
plot_ly(x = time(USAccDeaths), y = USAccDeaths) %>%
  add_lines() %>%
  rangeslider()
```

---

schema *Display plotly's plot schema*

Description

The schema contains valid attributes names, their value type, default values (if any), and min/max values (if applicable).

Usage

```
schema()
```

Examples

```
schema()
```

---

signup *Create a new plotly account.*

Description

A sign up interface to plotly through the R Console.

Usage

```
signup(username, email, save = TRUE)
```

Arguments

username	Desired username.
email	Desired email.
save	If request is successful, should the username & API key be automatically stored as an environment variable in a .Rprofile?

Value

- api_key key to use with the api
- tmp_pw temporary password to access your plotly account

References

https://plot.ly/rest/

Examples

```
## Not run:
# You need a plotly username and API key to communicate with the plotly API.

# If you don't already have an API key, you can obtain one with a valid
# username and email via signup().
s <- signup('anna.lyst', 'anna.lyst@plot.ly')

# If you already have a username and API key, please create the following
# environment variables:
Sys.setenv("plotly_username" = "me")
Sys.setenv("plotly_api_key" = "mykey")
# You can also change the default domain if you have a plotly server.
Sys.setenv("plotly_domain" = "http://mydomain.com")

# If you want to automatically load these environment variables when you
# start R, you can put them inside your ~/.Rprofile
# (see help(.Rprofile) for more details)


## End(Not run)
```

---

style *Modify trace(s)*

Description

Modify trace(s) of an existing plotly visualization. Useful when used in conjunction with `get_figure()`.

Usage

```
style(p, ..., traces = 1)
```

Arguments

p	A plotly visualization.
...	Visual properties.
traces	numeric vector. Which traces should be modified?

Author(s)

Carson Sievert

See Also

get_figure()

Examples

```
p <- qplot(data = mtcars, wt, mpg, geom = c("point", "smooth"))
# keep the hover info for points, but remove it for the line/ribbon
style(p, hoverinfo = "none", traces = c(2, 3))
```

---

subplot *View multiple plots in a single view*

Description

View multiple plots in a single view

Usage

```
subplot(..., nrows = 1, widths = NULL, heights = NULL, margin = 0.02,
  shareX = FALSE, shareY = FALSE, titleX = shareX, titleY = shareY,
  which_layout = "merge")
```

Arguments

...	One of the following • any number of plotly/ggplot2 objects. • a list of plotly/ggplot2 objects. • a tibble with one list-column of plotly/ggplot2 objects.
nrows	number of rows for laying out plots in a grid-like structure. Only used if no domain is already specified.
widths	relative width of each column on a 0-1 scale. By default all columns have an equal relative width.
heights	relative height of each row on a 0-1 scale. By default all rows have an equal relative height.

margin	either a single value or four values (all between 0 and 1). If four values are provided, the first is used as the left margin, the second is used as the right margin, the third is used as the top margin, and the fourth is used as the bottom margin. If a single value is provided, it will be used as all four margins.
shareX	should the x-axis be shared amongst the subplots?
shareY	should the y-axis be shared amongst the subplots?
titleX	should x-axis titles be retained?
titleY	should y-axis titles be retained?
which_layout	adopt the layout of which plot? If the default value of "merge" is used, layout options found later in the sequence of plots will override options found earlier in the sequence. This argument also accepts a numeric vector specifying which plots to consider when merging.

Value

A plotly object

Author(s)

Carson Sievert

Examples

```
# pass any number of plotly objects to subplot()
p1 <- plot_ly(economics, x = ~date, y = ~uempmed)
p2 <- plot_ly(economics, x = ~date, y = ~unemploy)
subplot(p1, p2, p1, p2, nrows = 2, margin = 0.05)

# or pass a list
library(purrr)
economics_long %>%
  split(.$variable) %>%
  map(~ plot_ly(., x = ~date, y = ~value)) %>%
  subplot(nrows = NROW(.), shareX = TRUE)

# or pass a tibble with a list-column of plotly objects
economics_long %>%
  group_by(variable) %>%
  do(p = plot_ly(., x = ~date, y = ~value)) %>%
  subplot(nrows = NROW(.), shareX = TRUE)

# learn more at https://cpsievert.github.io/plotly_book/subplot.html
```

toRGB *Convert R colours to RGBA hexadecimal colour values*

Description

Convert R colours to RGBA hexadecimal colour values

Usage

```
toRGB(x, alpha = 1)
```

Arguments

x	see the col argument in col2rgb for valid specifications
alpha	alpha channel on 0-1 scale

Value

hexadecimal colour value (if is.na(x), return "transparent" for compatibility with Plotly)

toWebGL *Convert trace types to WebGL*

Description

Convert trace types to WebGL

Usage

```
toWebGL(p)
```

Arguments

p	a plotly or ggplot object.

Examples

```
# currently no bargl trace type
toWebGL(qplot(1:10))
toWebGL(qplot(1:10, 1:10))
```

to_basic *Convert a geom to a "basic" geom.*

Description

This function makes it possible to convert ggplot2 geoms that are not included with ggplot2 itself. Users shouldn't need to use this function. It exists purely to allow other package authors to write their own conversion method(s).

Usage

```
to_basic(data, prestats_data, layout, params, p, ...)
```

Arguments

data	the data returned by ggplot2::ggplot_build().
prestats_data	the data before statistics are computed.
layout	the panel layout.
params	parameters for the geom, statistic, and 'constant' aesthetics
p	a ggplot2 object (the conversion may depend on scales, for instance).
...	currently ignored

wind *Wind data*

Description

Description TBD.

Usage

```
wind
```

Format

A data frame with three variables: r, t, nms.

Index