

Package ‘trelliscopejs’

May 8, 2026

Title Create Interactive Trelliscope Displays

Version 0.2.11

Description Trelliscope is a scalable, flexible, interactive approach to visualizing data (Hafen, 2013 <[doi:10.1109/LDAV.2013.6675164](https://doi.org/10.1109/LDAV.2013.6675164)>). This package provides methods that make it easy to create a Trelliscope display specification for TrelliscopeJS. High-level functions are provided for creating displays from within 'tidyverse' or 'ggplot2' workflows. Low-level functions are also provided for creating new interfaces.

Depends R (>= 3.4.0)

License BSD_3_clause + file LICENSE

Encoding UTF-8

LazyData true

Imports dplyr, purrr, grid, htmltools, DistributionUtils, grDevices, gtable, digest, jsonlite, ggplot2 (>= 3.2.1), base64enc, htmlwidgets, graphics, progress, utils, knitr, webshot, autocogs, tidyr, rlang, fidelius

Suggests plotly, testthat, covr, gapminder, rmarkdown, shiny, housingData

RoxygenNote 7.3.2

URL <https://hafen.github.io/trelliscopejs/>,
<https://github.com/hafen/trelliscopejs>

BugReports <https://github.com/hafen/trelliscopejs/issues>

VignetteBuilder knitr

NeedsCompilation no

Author Ryan Hafen [aut, cre] (ORCID: <<https://orcid.org/0000-0002-5516-8367>>),
Barret Schloerke [aut]

Maintainer Ryan Hafen <rhafen@gmail.com>

Repository CRAN

Date/Publication 2025-07-23 02:10:02 UTC

Contents

as_cognostics	2
cog	3
cogs	5
cog_disp_filter	6
cog_href	7
create_cog_template	8
facet_trelliscope	8
img_panel	11
img_panel_local	12
input_cogs	12
input_cogs_api	13
input_radio	13
input_text	14
map2_cog	15
map2_plot	16
map_cog	17
map_plot	18
md_description	19
mpg_labels	19
panels	20
prepare_display	20
print.facet_trelliscope	21
set_labels	22
sort_spec	22
trelliscope	23
Trelliscope-shiny	26
tr_charm	27
update_display_list	27
view_item	28
view_list	28
write_cognostics	29
write_config	29
write_display_obj	30
write_panel	32
write_panels	33
Index	34

as_cognostics

Cast a data frame as a cognostics data frame

Description

Cast a data frame as a cognostics data frame

Usage

```
as_cognostics(
  x,
  cond_cols,
  key_col = NULL,
  cog_desc = NULL,
  needs_key = TRUE,
  needs_cond = TRUE,
  group = "common"
)
```

Arguments

x	a data frame
cond_cols	the column name(s) that comprise the conditioning variables
key_col	the column name that indicates the panel key
cog_desc	an optional named list of descriptions for the cognostics columns
needs_key	does the result need to have a "key" column?
needs_cond	does the result need to have conditioning variable columns?
group	value to be used in the <code>cog</code> group

cog

Cast Column as a Cognostic

Description

Cast a column of a cognostics data frame as a cognostic object

Usage

```
cog(
  val = NULL,
  desc = "",
  group = "common",
  type = NULL,
  default_label = FALSE,
  default_active = TRUE,
  filterable = TRUE,
  sortable = TRUE,
  log = NULL
)
```

Arguments

<code>val</code>	a scalar value (numeric, character, date, etc.)
<code>desc</code>	a description for this cognostic value
<code>group</code>	optional categorization of the cognostic for organizational purposes in the viewer (currently not implemented in the viewer)
<code>type</code>	the desired type of cognostic you would like to compute (see details)
<code>default_label</code>	should this cognostic be used as a panel label in the viewer by default?
<code>default_active</code>	should this cognostic be active (available for sort / filter / sample) by default?
<code>filterable</code>	should this cognostic be filterable? Default is TRUE. It can be useful to set this to FALSE if the cognostic is categorical with many unique values and is only desired to be used as a panel label.
<code>sortable</code>	should this cognostic be sortable?
<code>log</code>	when being used in the viewer for visual univariate and bivariate filters, should the log be computed? Useful when the distribution of the cognostic is very long-tailed or has large outliers. Can either be a logical or a positive integer indicating the base.

Details

Different types of cognostics can be specified through the `type` argument that will affect how the user is able to interact with those cognostics in the viewer. This can usually be ignored because it will be inferred from the implicit data type of `val`. But there are special types of cognostics, such as geographic coordinates and relations (not implemented) that can be specified as well. Current possibilities for `type` are "key", "integer", "numeric", "factor", "date", "time", "href".

Value

object of class "cog"

Examples

```
library(dplyr)
library(tidyr)
library(purrr)
library(ggplot2)
library(plotly)

mpg_cog <- mpg %>%
  nest(data = !one_of(c("manufacturer", "class"))) %>%
  mutate(
    cogs = map_cog(data, ~ tibble(
      mean_city_mpg = cog(mean(.$cty), desc = "Mean city mpg"),
      mean_hwy_mpg = cog(mean(.$hwy), desc = "Mean highway mpg"),
      most_common_drv = cog(tail(names(table(.$drv)), 1), desc = "Most common drive type")
    )),
    panel = map_plot(data, function(x) {
      plot_ly(data = x, x = ~cty, y = ~hwy,
        type = "scatter", mode = "markers") %>%
```

```

      layout(
        xaxis = list(range = c(9, 47)),
        yaxis = list(range = c(7, 37)))
    })
  )

  trelliscope(mpg_cog, name = "city_vs_highway_mpg", nrow = 1, ncol = 2)

```

cogs

Cogs Wrapper Function

Description

Cogs Wrapper Function

Usage

```
cogs(.x, .f, ...)
```

Arguments

`.x` a list or atomic vector (see [map](#) for details)
`.f` a function, formula, or atomic vector (see [map](#) for details)
`...` additional arguments passed on to `.f` (see [map](#) for details)

Details

See [map](#)

Examples

```

library(dplyr)
library(tidyr)
library(plotly)
ggplot2::mpg %>%
  nest(data = !one_of(c("manufacturer", "class"))) %>%
  mutate(
    additional_cogs = map_cog(data, function(x) {
      tibble(
        max_city_mpg = cog(max(x$cty), desc = "Max city mpg"),
        min_city_mpg = cog(min(x$cty), desc = "Min city mpg")
      )
    }),
    panel = map_plot(data, function(x) {
      plot_ly(data = x, x = ~cty, y = ~hwy,
        type = "scatter", mode = "markers")
    })
  ) %>%
  trelliscope(name = "city_vs_highway_mpg", nrow = 1, ncol = 2)

```

cog_disp_filter	<i>Helper function for creating a cognostic for a link to another display in a filtered state</i>
-----------------	---

Description

Helper function for creating a cognostic for a link to another display in a filtered state

Usage

```
cog_disp_filter(
  display,
  var,
  val,
  desc = "link",
  group = "common",
  type = c("href", "href_hash"),
  default_label = FALSE,
  default_active = FALSE,
  filterable = FALSE,
  sortable = FALSE
)
```

Arguments

display	A string indicating the name of the display to link to.
var	A string indicating the variable name to filter on.
val	A string indicating the value of the filter.
desc	a description for this cognostic value
group	optional categorization of the cognostic for organizational purposes in the viewer (currently not implemented in the viewer)
type	of either "href" or "href_hash". "href" will open the link in a new page. "href_hash" will update the page's hash and reload the page (useful when changing state inside an iframe)
default_label	should this cognostic be used as a panel label in the viewer by default?
default_active	should this cognostic be active (available for sort / filter / sample) by default?
filterable	should this cognostic be filterable? Default is TRUE. It can be useful to set this to FALSE if the cognostic is categorical with many unique values and is only desired to be used as a panel label.
sortable	should this cognostic be sortable?

cog_href

Href Cognostic

Description

Create href to be used as cognostics in a trelliscope display

Usage

```
cog_href(
  x,
  desc = "link",
  group = "common",
  type = c("href", "href_hash"),
  default_label = FALSE,
  default_active = FALSE,
  filterable = FALSE,
  sortable = FALSE,
  log = FALSE
)
```

Arguments

`x` URL to link to
`desc`, `group`, `default_label`, `default_active`, `filterable`, `sortable`, `log`
arguments passed to [cog](#)

`type` of either "href" or "href_hash". "href" will open the link in a new page. "href_hash" will update the page's hash and reload the page (useful when changing state inside an iframe)

See Also

[cog](#)

Examples

```
library(dplyr)
library(tidyr)
library(plotly)
iris %>%
  nest(data = ~Species) %>%
  mutate(
    panel = map_plot(data, function(x) {
      plot_ly(data = x, x = ~Sepal.Length, y = ~Sepal.Width,
        type = "scatter", mode = "markers")
    })
  ),
  wiki_link = cog_href(paste0("https://en.wikipedia.org/wiki/Iris_",
    tolower(Species))[1], default_label = TRUE,
```

```

      desc = "link to species on wikipedia")
    ) %>%
  trelliscope(name = "iris_species", ncol = 3)

```

create_cog_template	<i>Create a cognostics template that can be edited and used to specify cognostics in a display</i>
---------------------	--

Description

Create a cognostics template that can be edited and used to specify cognostics in a display

Usage

```
create_cog_template(x = NULL)
```

Arguments

x a data frame that will be used as an input for the trelliscope display. If NULL, a blank template will be created.

Note

The input 'x' can be a starting point and does not need to contain all variables that may be added later. Also note that after you edit the output (for example using the editData package, or writing to a csv and editing in Excel) to reflect how you would like the cognostics to display in the viewer, you can Add this specification to your trelliscope display with [add_cog_template()].

facet_trelliscope	<i>Facet Trelliscope</i>
-------------------	--------------------------

Description

Facet Trelliscope

Usage

```

facet_trelliscope(
  facets,
  nrow = 1,
  ncol = 1,
  scales = "same",
  name = NULL,
  group = "common",
  desc = ggplot2::waiver(),

```

```

md_desc = ggplot2::waiver(),
path = NULL,
height = 500,
width = 500,
inputs = NULL,
state = NULL,
views = NULL,
jsonp = TRUE,
as_plotly = FALSE,
plotly_args = NULL,
plotly_cfg = NULL,
split_sig = NULL,
google_analytics_id = NULL,
self_contained = FALSE,
thumb = TRUE,
require_token = FALSE,
id = NULL,
order = 1,
disclaimer = FALSE,
update_plots = TRUE,
auto_cog = FALSE,
split_layout = FALSE,
data = ggplot2::waiver()
)

```

Arguments

facets	formula to facet the panels on. Similar to <code>ggplot2::facet_wrap</code> 's facets
nrow	the number of rows of panels to display by default
ncol	the number of columns of panels to display by default
scales	should scales be the same ("same", the default), free ("free"), or sliced ("sliced"). May provide a single string or two strings, one for the X and Y axis respectively.
name	name of the display
group	group that the display belongs to
desc	description of the display
md_desc	optional string of markdown that will be shown in the viewer for additional context about the display
path	the base directory of the trelliscope application
height	height in pixels of each panel
width	width in pixels of each panel
inputs	optional set of input specifications (using <code>input_cogs</code>) to allow user input for each panel
state	the initial state the display will open in
views	an optional list of pre-specified views of the display (experimental)
jsonp	should json for display object be jsonp (TRUE) or json (FALSE)?

<code>as_plotly</code>	should the panels be written as plotly objects?
<code>plotly_args</code>	optional named list of arguments to send to ggplotly
<code>plotly_cfg</code>	optional named list of arguments to send to plotly's config method
<code>split_sig</code>	optional string that specifies the "signature" of the data splitting. If not specified, this is calculated as the md5 hash of the sorted unique facet variables. This is used to identify "related displays" - different displays that are based on the same faceting scheme. This parameter should only be specified manually if a display's faceting is mostly similar to another display's.
<code>google_analytics_id</code>	optional string specifying Google Analytics ID
<code>self_contained</code>	should the Trelliscope display be a self-contained html document? (see note)
<code>thumb</code>	should a thumbnail be created?
<code>require_token</code>	require a special token for all displays to be visible (experimental)
<code>id</code>	set a hard-coded ID for this app (do not set this if the display will be part of a larger web page)
<code>order</code>	an integer indicating the order that the display should appear in if using multiple displays
<code>disclaimer</code>	an optional string of html to include as a disclaimer for the set of displays
<code>update_plots</code>	should the plots be updated? This is to allow slight updates to the underlying display data without the need to re-render all of the plots. Use it carefully.
<code>auto_cog</code>	should auto cogs be computed (if possible)?
<code>split_layout</code>	boolean that determines if the layout is split into components like a <code>facet_grid</code> vs. individual panels like <code>facet_wrap</code> . Only applies to <code>ggplot2</code> plot objects.
<code>data</code>	data used for faceting. Defaults to the first layer data

Note

Note that `self_contained` is severely limiting and should only be used in cases where you would either like your display to show up in the RStudio viewer pane, in an interactive R Markdown Notebook, or in a self-contained R Markdown html document.

Note that `self_contained` is severely limiting and should only be used in cases where you would either like your display to show up in the RStudio viewer pane, in an interactive R Markdown Notebook, or in a self-contained R Markdown html document.

Examples

```
## Not run:
library(ggplot2)

# basically swap out facet_wrap for facet_trelliscope
qplot(cty, hwy, data = mpg) +
  facet_trelliscope(~ class + manufacturer)

# not required, but if you set labels, these will be added as
# descriptions to the cognostics that are automatically computed
```

```
mpg <- set_labels(mpg, mpg_labels)

qplot(cty, hwy, data = mpg) +
  theme_bw() +
  facet_trelliscope(~ manufacturer + class, nrow = 2, ncol = 4)

# using plotly
library(plotly)
qplot(cty, hwy, data = mpg) +
  theme_bw() +
  facet_trelliscope(~ manufacturer + class, nrow = 2, ncol = 4, as_plotly = TRUE)

qplot(class, cty, data = mpg, geom = c("boxplot", "jitter"), na.rm = TRUE) +
  facet_trelliscope(~ class, ncol = 7, height = 800, width = 200,
    state = list(sort = list(sort_spec("cty_mean")))) +
  theme_bw()

library(gapminder)
qplot(year, lifeExp, data = gapminder) +
  xlim(1948, 2011) + ylim(10, 95) + theme_bw() +
  facet_trelliscope(~ country + continent, nrow = 2, ncol = 7,
    width = 300, as_plotly = TRUE,
    plotly_cfg = list(displayModeBar = FALSE))

## End(Not run)
```

img_panel

Cast a vector of URLs pointing to images as an image panel source

Description

Cast a vector of URLs pointing to images as an image panel source

Usage

```
img_panel(x)
```

Arguments

x a vector of URLs pointing to images

img_panel_local	<i>Cast a vector of URLs pointing to local images as an image panel source</i>
-----------------	--

Description

Cast a vector of URLs pointing to local images as an image panel source

Usage

```
img_panel_local(x)
```

Arguments

x a vector of URLs pointing to images

Note

x must be paths relative to the path argument passed to [trelliscope](#).

Examples

```
## Not run:
# assuming images are available locally in relative path pokemon_local/images
pokemon$img <- img_panel_local(paste0("images/", basename(pokemon$url_image)))
trelliscope(pokemon, name = "pokemon", path = "pokemon_local")

## End(Not run)
```

input_cogs	<i>Specify a collection of input cognostics to be stored in browser local-Storage</i>
------------	---

Description

Specify a collection of input cognostics to be stored in browser localStorage

Usage

```
input_cogs(..., feedback_email = NULL, extra_cogs = NULL)
```

Arguments

feedback_email optional feedback email address that input feedback can be sent to
 extra_cogs optional vector of names of non-input "regular" cognostics to include in the csv
 output
 ... objects created by any of [input_radio](#), [input_text](#)

input_cogs_api	<i>Specify a collection of input cognostics to be stored using an API</i>
----------------	---

Description

Specify a collection of input cognostics to be stored using an API

Usage

```
input_cogs_api(
    ...,
    set_url,
    get_url,
    get_request_options = list(mode = "cors", method = "GET", headers = list(`Content-Type`
    = "application/json", Accept = "application/json")),
    set_request_options = list(mode = "cors", method = "POST", headers =
    list(`Content-Type` = "application/json", Accept = "application/json"))
)
```

Arguments

set_url	URL of the API endpoint for setting a single input
get_url	URL of the API endpoint for getting all inputs for the display
get_request_options	request options for the API call to set inputs
set_request_options	request options for the API call to get inputs
...	objects created by any of input_radio , input_text

Details

See [\[here\]\(https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch#supplying_request_options\)](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch#supplying_request_options) for more information about request options.

input_radio	<i>Specify a radio button input</i>
-------------	-------------------------------------

Description

Specify a radio button input

Usage

```
input_radio(name, desc = NULL, options, group = NULL, default_label = TRUE)
```

Arguments

name	name of the input
desc	optional description of the input
options	a character vector of options to select between
group	optional categorization of the input for organizational purposes in the viewer (currently not implemented in the viewer)
default_label	should this input be shown under the panel in the viewer by default?

input_text	<i>Specify a text input</i>
------------	-----------------------------

Description

Specify a text input

Usage

```
input_text(
  name,
  desc = NULL,
  width = 80,
  height = 3,
  group = NULL,
  default_label = TRUE
)
```

Arguments

name	name of the input
desc	optional description of the input
width	width (in characters) of the text box popout
height	height (in lines of text) of the text box popout
group	optional categorization of the input for organizational purposes in the viewer (currently not implemented in the viewer)
default_label	should this input be shown under the panel in the viewer by default?

map2_cog	<i>Map over multiple inputs simultaneously and return a vector of cognostics data frames</i>
----------	--

Description

Map over multiple inputs simultaneously and return a vector of cognostics data frames

Usage

```
map2_cog(.x, .y, .f, ...)
```

```
pmap_cog(.l, .f, ...)
```

Arguments

.x, .y	Vectors of the same length. A vector of length 1 will be recycled.
.f	A function, formula, or atomic vector (see map2 for details)
...	additional arguments passed on to .f.
.l	A list of lists. The length of .l determines the number of arguments that .f will be called with. List names will be used if present.

Details

See [map2](#)

Examples

```
library(tidyr)
library(purrr)
library(plotly)
library(dplyr)

iris %>%
  nest(data = ~Species) %>%
  mutate(
    mod = map(data, ~ lm(Sepal.Length ~ Sepal.Width, data = .x)),
    cogs = map2_cog(data, mod, function(data, mod) {
      tibble(max_sl = max(data$Sepal.Length), slope = coef(mod)[2])
    }),
    panel = map2_plot(data, mod, function(data, mod) {
      plot_ly(data = data, x = ~Sepal.Width, y = ~Sepal.Length,
        type = "scatter", mode = "markers", name = "data") %>%
        add_trace(data = data, x = ~Sepal.Width, y = ~predict(mod),
          mode = "lines", name = "lm")
    }) %>%
    trelliscope(name = "iris")
```

 map2_plot

Map over multiple inputs simultaneously and return a vector of plots

Description

Map over multiple inputs simultaneously and return a vector of plots

Usage

```
map2_plot(.x, .y, .f, ...)
```

```
pmap_plot(.l, .f, ...)
```

Arguments

<code>.x, .y</code>	Vectors of the same length. A vector of length 1 will be recycled.
<code>.f</code>	A function, formula, or atomic vector (see map2 for details)
<code>...</code>	additional arguments passed on to <code>.f</code> .
<code>.l</code>	A list of lists. The length of <code>.l</code> determines the number of arguments that <code>.f</code> will be called with. List names will be used if present.

Details

See [map2](#)

Examples

```
library(tidyr)
library(purrr)
library(plotly)
library(dplyr)

iris %>%
  nest(data = -Species) %>%
  mutate(
    mod = map(data, ~ lm(Sepal.Length ~ Sepal.Width, data = .x)),
    panel = map2_plot(data, mod, function(data, mod) {
      plot_ly(data = data, x = ~Sepal.Width, y = ~Sepal.Length,
        type = "scatter", mode = "markers", name = "data") %>%
      add_trace(data = data, x = ~Sepal.Width, y = ~predict(mod),
        mode = "lines", name = "lm")
    }) %>%
  )
  trelliscope(name = "iris")
```

map_cog	<i>Apply a function to each element of a vector and return a vector of cognostics data frames</i>
---------	---

Description

Apply a function to each element of a vector and return a vector of cognostics data frames

Usage

```
map_cog(.x, .f, ...)
```

Arguments

.x	a list or atomic vector (see map for details)
.f	a function, formula, or atomic vector (see map for details)
...	additional arguments passed on to .f (see map for details)

Details

See [map](#)

Examples

```
library(dplyr)
library(tidyr)
library(plotly)
ggplot2::mpg %>%
  nest(data = !one_of(c("manufacturer", "class"))) %>%
  mutate(
    cog = map_cog(data, function(x) tibble(mean_hwy = mean(x$hwy))),
    panel = map_plot(data, function(x) {
      plot_ly(data = x, x = ~cty, y = ~hwy,
        type = "scatter", mode = "markers")
    })
  ) %>%
  trelliscope(name = "city_vs_highway_mpg")
```

map_plot	<i>Apply a function to each element of a vector and return a vector of plots</i>
----------	--

Description

Apply a function to each element of a vector and return a vector of plots

Usage

```
map_plot(.x, .f, ...)
```

Arguments

.x	a list or atomic vector (see map for details)
.f	a function, formula, or atomic vector (see map for details)
...	additional arguments passed on to .f (see map for details)

Details

See [map](#)

Examples

```
library(dplyr)
library(tidyr)
library(purrr)
library(plotly)
library(gapminder)

# nest gapminder data by country
by_country <- gapminder %>%
  nest(data = !one_of(c("country", "continent")))

# add in a plot column with map_plot
by_country <- by_country %>% mutate(
  panel = map_plot(data, function(x) {
    plot_ly(data = x, x = ~year, y = ~lifeExp,
            type = "scatter", mode = "markers") %>%
    layout(
      xaxis = list(range = c(1948, 2011)),
      yaxis = list(range = c(10, 95)))
  })
)

# plot it
by_country %>%
  trelliscope("gapminder", nrow = 2, ncol = 7, width = 300)

# example using mpg data
```

```
ggplot2::mpg %>%
  nest(data = !one_of(c("manufacturer", "class"))) %>%
  mutate(panel = map_plot(data, function(x) {
    plot_ly(data = x, x = ~hwy, y = ~cty,
      type = "scatter", mode = "markers")
  }))) %>%
  trelliscope(name = "city_vs_highway_mpg")
```

md_description	<i>Specify a markdown description for a display</i>
----------------	---

Description

Specify a markdown description for a display

Usage

```
md_description(
  content = "",
  title = "Information About This Display",
  show = FALSE
)
```

Arguments

content	Markdown content
title	Title of the dialog box that displays this content
show	should the markdown description be shown by default when the display is loaded?

mpg_labels	<i>Labels for ggplot2 "mpg" data</i>
------------	--------------------------------------

Description

Labels for ggplot2 "mpg" data

Usage

```
mpg_labels
```

Format

An object of class list of length 10.

panels

Panels Wrapper Function

Description

Panels Wrapper Function

Usage

```
panels(.x, .f, ...)
```

Arguments

`.x` a list or atomic vector (see [map](#) for details)
`.f` a function, formula, or atomic vector (see [map](#) for details)
`...` additional arguments passed on to `.f` (see [map](#) for details)

Details

See [map](#)

Examples

```
library(dplyr)
library(tidyr)
library(plotly)
ggplot2::mpg %>%
  nest(data = !one_of(c("manufacturer", "class"))) %>%
  mutate(panel = map_plot(data, function(x) {
    plot_ly(data = x, x = ~hwy, y = ~cty,
             type = "scatter", mode = "markers")
  })) %>%
  trelliscope(name = "city_vs_highway_mpg")
```

prepare_display

Set up all auxiliary files needed for a Trelliscope app

Description

Set up all auxiliary files needed for a Trelliscope app

Usage

```
prepare_display(
  base_path,
  id,
  self_contained = FALSE,
  jsonp = TRUE,
  require_token = FALSE,
  disclaimer = FALSE,
  pb = NULL
)
```

Arguments

base_path	the base directory of the trelliscope application
id	a unique id for the application
self_contained	should the Trelliscope display be a self-contained html document?
jsonp	should json for display list and app config be jsonp (TRUE) or json (FALSE)?
require_token	require a special token for all displays to be visible (experimental)
disclaimer	an optional string of html to include as a disclaimer for the set of displays
pb	optional progress bar object to pass in and use to report progress

```
print.facet_trelliscope
```

Print facet trelliscope object

Description

Print facet trelliscope object

Usage

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

Arguments

x	plot object
...	ignored

set_labels	<i>Set labels for a data frame</i>
------------	------------------------------------

Description

Set labels for a data frame

Usage

```
set_labels(dat, label_list)
```

Arguments

dat	a data frame to apply labels to
label_list	a named list with names matching those of dat and values being labels

Value

data frame with labels attached as attributes (attached to each column and named "label")

sort_spec	<i>Specify how a display should be sorted</i>
-----------	---

Description

Specify how a display should be sorted

Usage

```
sort_spec(name, dir = "asc")
```

Arguments

name	variable name to sort on
dir	direction to sort ('asc' or 'desc')

`trelliscope`*Create a Trelliscope Display*

Description

Create a Trelliscope Display

Usage

```
trelliscope(  
  x,  
  name,  
  group = "common",  
  panel_col = NULL,  
  cond_cols = NULL,  
  desc = "",  
  md_desc = "",  
  path,  
  height = 500,  
  width = 500,  
  inputs = NULL,  
  google_analytics_id = NULL,  
  auto_cog = FALSE,  
  state = NULL,  
  views = NULL,  
  nrow = 1,  
  ncol = 1,  
  jsonp = TRUE,  
  split_sig = NULL,  
  self_contained = FALSE,  
  thumb = FALSE,  
  require_token = FALSE,  
  id = NULL,  
  order = 1,  
  disclaimer = FALSE,  
  update_plots = TRUE  
)
```

Arguments

<code>x</code>	an object to create at trelliscope display for
<code>name</code>	name of the display
<code>group</code>	group that the display belongs to
<code>panel_col</code>	optional string specifying the column to use for panels (if there are multiple plot columns in <code>x</code>)

<code>cond_cols</code>	optional vector of columns to specify as "conditioning" columns - combined they must be unique and not have any missing values
<code>desc</code>	optional text description of the display
<code>md_desc</code>	optional string of markdown that will be shown in the viewer for additional context about the display
<code>path</code>	the base directory of the trelliscope application
<code>height</code>	height in pixels of each panel
<code>width</code>	width in pixels of each panel
<code>inputs</code>	optional set of input specifications (using <code>input_cogs</code>) to allow user input for each panel
<code>google_analytics_id</code>	optional string specifying Google Analytics ID
<code>auto_cog</code>	should auto cogs be computed (if possible)?
<code>state</code>	the initial state the display will open in
<code>views</code>	an optional list of pre-specified views of the display (experimental)
<code>nrow</code>	the number of rows of panels to display by default
<code>ncol</code>	the number of columns of panels to display by default
<code>jsonp</code>	should json for display object be jsonp (TRUE) or json (FALSE)?
<code>split_sig</code>	optional string that specifies the "signature" of the data splitting. If not specified, this is calculated as the md5 hash of the sorted unique facet variables. This is used to identify "related displays" - different displays that are based on the same faceting scheme. This parameter should only be specified manually if a display's faceting is mostly similar to another display's.
<code>self_contained</code>	should the Trelliscope display be a self-contained html document? (see note)
<code>thumb</code>	should a thumbnail be created?
<code>require_token</code>	require a special token for all displays to be visible (experimental)
<code>id</code>	set a hard-coded ID for this app (do not set this if the display will be part of a larger web page)
<code>order</code>	an integer indicating the order that the display should appear in if using multiple displays
<code>disclaimer</code>	an optional string of html to include as a disclaimer for the set of displays
<code>update_plots</code>	should the plots be updated? This is to allow slight updates to the underlying display data without the need to re-render all of the plots. Use it carefully.

Note

Note that `self_contained` is severely limiting and should only be used in cases where you would either like your display to show up in the RStudio viewer pane, in an interactive R Markdown Notebook, or in a self-contained R Markdown html document.

Examples

```

## Not run:
library(dplyr)
library(tidyr)
library(purrr)
library(plotly)
library(ggplot2)

# tidyverse + plotly
d <- mpg %>%
  nest(data = !one_of(c("manufacturer", "class"))) %>%
  mutate(
    mean_city_mpg = map_dbl(data, ~ mean(.$cty)),
    panel = map_plot(data, function(x) {
      plot_ly(data = x, x = ~cty, y = ~hwy,
              type = "scatter", mode = "markers")
    })
  )

d %>% trelliscope(name = "city_vs_highway_mpg")

# set default layout
d %>% trelliscope(name = "city_vs_highway_mpg", nrow = 2, ncol = 3)

# set the output path for where files will be stored
my_displays <- tempfile()
d %>% trelliscope(name = "city_vs_highway_mpg", path = my_displays)

# multiple displays can be added to the same path and all will be available in the viewer
d %>% trelliscope(name = "city_vs_highway_mpg2", path = my_displays)

# ordering the data frame will set default sort order of the display
d %>%
  arrange(-mean_city_mpg) %>%
  trelliscope(name = "city_vs_highway_mpg")

# tidyverse + ggplot2
mpg %>%
  nest(data = !one_of(c("manufacturer", "class"))) %>%
  mutate(
    panel = map_plot(data, ~
      qplot(cty, hwy, data = .) + xlab("cty") + ylab("hwy") +
      xlim(7, 37) + ylim(9, 47) + theme_bw()) %>%
  trelliscope(name = "tidy_gg")

# computing additional cognostics
mpg_cog <- mpg %>%
  nest(data = !one_of(c("manufacturer", "class"))) %>%
  mutate(
    cogs = map_cog(data, ~ tibble(
      mean_city_mpg = mean(.$cty),
      mean_hwy_mpg = mean(.$hwy),

```

```

      most_common_drv = tail(names(table(.drv)), 1)
    ))
  )

# computing additional cognostics explicitly using cog()
# so we can specify descriptions, etc.
mpg_cog2 <- mpg %>%
  nest(data = !one_of(c("manufacturer", "class"))) %>%
  mutate(
    cogs = map_cog(data, ~ tibble(
      mean_city_mpg = cog(mean(.cty), desc = "Mean city mpg"),
      mean_hwy_mpg = cog(mean(.hwy), desc = "Mean highway mpg"),
      most_common_drv = cog(tail(names(table(.drv)), 1), desc = "Most common drive type")
    )),
    panel = map_plot(data, function(x) {
      plot_ly(data = x, x = ~cty, y = ~hwy,
        type = "scatter", mode = "markers") %>%
      layout(
        xaxis = list(range = c(9, 47)),
        yaxis = list(range = c(7, 37)))
    })
  )

mpg_cog2 %>%
  trelliscope(name = "city_vs_highway_mpg", nrow = 1, ncol = 2)

## End(Not run)

```

Description

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

Usage

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

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

Arguments

<code>outputId</code>	output variable to read from
<code>width, height</code>	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
<code>expr</code>	An expression that generates a <code>trelliscopejs_widget</code>
<code>env</code>	The environment in which to evaluate <code>expr</code> .

quoted Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

tr_charm *Use fidelius to password protect a trelliscope display*

Description

Use fidelius to password protect a trelliscope display

Usage

```
tr_charm(x, ...)
```

Arguments

x an object of class "facet_trelliscope" or "trelliscopejs_widget"
 ... arguments passed to [fidelius::charm()]

update_display_list *Update Trelliscope app display list file*

Description

Update Trelliscope app display list file

Usage

```
update_display_list(base_path, jsonp = TRUE)
```

Arguments

base_path the base directory of the trelliscope application
 jsonp should json for display list be jsonp (TRUE) or json (FALSE)?

view_item	<i>Construct a list of view items</i>
-----------	---------------------------------------

Description

Construct a list of view items

Usage

```
view_item(name, hash)
```

Arguments

name	A string indicating the name of the view (will be displayed in the "Views" sidebar)
hash	A URL hash that sends the user to this view (typically the URL starting with &nrow=...)

view_list	<i>Construct a list of view items</i>
-----------	---------------------------------------

Description

Construct a list of view items

Usage

```
view_list(...)
```

Arguments

...	Objects created using view_item
-----	---

write_cognostics	<i>Write cognostics data for a display in a Trelliscope app</i>
------------------	---

Description

Write cognostics data for a display in a Trelliscope app

Usage

```
write_cognostics(cogdf, base_path, id, name, group = "common", jsonp = TRUE)
```

Arguments

cogdf	a data frame of cognostics, prepared with as_cognostics
base_path	the base directory of the trelliscope application
id	a unique id for the application
name	name of the display
group	group that the display belongs to
jsonp	should json for cognostics be jsonp (TRUE) or json (FALSE)?

write_config	<i>Write Trelliscope app configuration file</i>
--------------	---

Description

Write Trelliscope app configuration file

Usage

```
write_config(  
  base_path,  
  id,  
  self_contained = FALSE,  
  jsonp = TRUE,  
  require_token = FALSE,  
  disclaimer = FALSE,  
  split_layout = FALSE,  
  has_legend = FALSE  
)
```

Arguments

base_path	the base directory of the trelliscope application
id	a unique id for the application
self_contained	should the Trelliscope display be a self-contained html document?
jsonp	should json for app config be jsonp (TRUE) or json (FALSE)?
require_token	require a special token for all displays to be visible (experimental)
disclaimer	an optional string of html to include as a disclaimer for the set of displays
split_layout	boolean that determines if the layout is split into components like a facet_grid vs. individual panels like facet_wrap. Only applies to ggplot2 plot objects.
has_legend	should a legend be reported for split_layout

write_display_obj	<i>Write a "display object" file for a Trelliscope app</i>
-------------------	--

Description

Write a "display object" file for a Trelliscope app

Usage

```
write_display_obj(
  cogdf,
  panel_example,
  base_path,
  id,
  name,
  group = "common",
  desc = "",
  height = 500,
  width = 500,
  inputs = NULL,
  md_desc = "",
  state = NULL,
  google_analytics_id = NULL,
  views = NULL,
  jsonp = TRUE,
  split_sig = NULL,
  panel_img_col = NULL,
  self_contained = FALSE,
  thumb = TRUE,
  split_layout = FALSE,
  split_aspect = NULL,
  has_legend = FALSE,
  order = 1,
  pb = NULL
)
```

Arguments

cogdf	a data frame of cognostics, prepared with as_cognostics
panel_example	an example object of one panel of a display (can be trellis, ggplot2, or htmlwidget object)
base_path	the base directory of the trelliscope application
id	a unique id for the application
name	name of the display
group	group that the display belongs to
desc	description of the display
height	height in pixels of each panel
width	width in pixels of each panel
inputs	optional set of input specifications (using input_cogs) to allow user input for each panel
md_desc	optional string of markdown that will be shown in the viewer for additional context about the display
state	the initial state the display will open in
google_analytics_id	optional string specifying Google Analytics ID
views	an optional list of pre-specified views of the display (experimental)
jsonp	should json for display object be jsonp (TRUE) or json (FALSE)?
split_sig	optional string "signature" specifying the data splitting
panel_img_col	which column (if any) is a panel image column?
self_contained	should the Trelliscope display be a self-contained html document?
thumb	should a thumbnail be created?
split_layout	boolean that determines if the layout is split into components like a <code>facet_grid</code> vs. individual panels like <code>facet_wrap</code> . Only applies to <code>ggplot2</code> plot objects.
split_aspect	list indicating aspect ratios of axes for a split layout. Only applies to <code>ggplot2</code> plot objects.
has_legend	should a legend be reported for <code>split_layout</code>
order	an integer indicating the order that the display should appear in if using multiple displays
pb	optional progress bar object to pass in and use to report progress

write_panel	<i>Write a plot object as a panel in a Trelliscope display</i>
-------------	--

Description

Write a plot object as a panel in a Trelliscope display

Usage

```
write_panel(  
  plot_object,  
  key,  
  base_path,  
  name,  
  group = "common",  
  width,  
  height,  
  jsonp = TRUE,  
  split_layout = FALSE  
)
```

Arguments

plot_object	a plot object to be written (can be trellis, ggplot2, or htmlwidget)
key	a string identifying the panel key, which will be used as the panel file name and which the panelKey column of the cognostics data frame should point to
base_path	the base directory of the trelliscope application
name	name of the display that the panel belongs to
group	group name of the display that the panel belongs to
width	width in pixels of each panel
height	height in pixels of each panel
jsonp	should json for panel be jsonp (TRUE) or json (FALSE)?
split_layout	boolean that determines if the layout is split into components like a facet_grid vs. individual panels like facet_wrap. Only applies to ggplot2 plot objects.

write_panels	<i>Write a list of plot objects as panels in a Trelliscope display</i>
--------------	--

Description

Write a list of plot objects as panels in a Trelliscope display

Usage

```
write_panels(plot_list, ..., pb = NULL)
```

Arguments

plot_list	a named list of plot objects to be written as panels (objects can be trellis, ggplot2, or htmlwidget) with the list names being the keys for the panels
...	params passed directly to write_panel
pb	optional progress bar object to pass in and use to report progress

Index

- * **datasets**
 - mpg_labels, 19
- as_cognostics, 2, 29, 31
- cog, 3, 3, 7
- cog_disp_filter, 6
- cog_href, 7
- cogs, 5
- create_cog_template, 8

- facet_trelliscope, 8
- facet_wrap, 9

- img_panel, 11
- img_panel_local, 12
- input_cogs, 9, 12, 24, 31
- input_cogs_api, 13
- input_radio, 12, 13, 13
- input_text, 12, 13, 14

- map, 5, 17, 18, 20
- map2, 15, 16
- map2_cog, 15
- map2_plot, 16
- map_cog, 17
- map_plot, 18
- md_description, 19
- mpg_labels, 19

- panels, 20
- pmap_cog (map2_cog), 15
- pmap_plot (map2_plot), 16
- prepare_display, 20
- print.facet_trelliscope, 21

- renderTrelliscope (Trelliscope-shiny), 26

- set_labels, 22
- sort_spec, 22

- tr_charm, 27
- trelliscope, 12, 23
- Trelliscope-shiny, 26
- trelliscopeOutput (Trelliscope-shiny), 26

- update_display_list, 27

- view_item, 28, 28
- view_list, 28

- write_cognostics, 29
- write_config, 29
- write_display_obj, 30
- write_panel, 32, 33
- write_panels, 33