Package 'd3po'

July 20, 2026

Type Package

Title Fast and Beautiful Interactive Visualization Based on the 'D3' Engine

Version 1.1.0

Description A fully FOSS alternative to 'Highcharter' based on 'D3' which provides functions for both fast and beautiful interactive visualization for 'Markdown' 'Shiny', and 'Tabler'.

Depends htmlwidgets, R (>= 4.1)

URL https://pacha.dev/d3po/

BugReports https://github.com/pachadotdev/d3po/issues

License Apache License (>= 2.0)

Encoding UTF-8

LazyData true

NeedsCompilation no

Suggests knitr, igraph, shiny, golem, geojsonsf, sf, jsonlite, rmarkdown, tintin

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Author Mauricio Vargas Sepulveda [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-1017-7574>),
John Coene [aut],
Ariel Alvarado [ctb],
Sylvain Lesage [ctb],
Curran Kelleher [ctb],
Fernando Becerra [ctb],
Natural Earth [dtc],
R Consortium [fnd] (Funded for the 2016-2017 ISC grants cycle)

Maintainer Mauricio Vargas Sepulveda <m.vargas.sepulveda@gmail.com>

Repository CRAN

Date/Publication 2026-07-20 12:20:37 UTC

Contents

d3po-exports . . . 2
d3po-shiny . . . 2
daes . . . 3
national . . . 4
po_area . . . 5
po_bar . . . 6
po_box . . . 7
po_donut . . . 8
po_download . . . 9
po_font . . . 9
po_format . . . 10
po_geomap . . . 10
po_labels . . . 11
po_line . . . 12
po_network . . . 13
po_pie . . . 15
po_scatter . . . 16
po_theme . . . 17
po_tooltip . . . 18
po_treemap . . . 18
subnational . . . 19
trade . . . 20

Index **21**

`d3po-exports` *D3po (re)exported methods*

Description

D3po (re)exported methods

`d3po-shiny` *Shiny bindings for 'd3po'*

Description

Output and render functions for using d3po within Shiny applications and interactive Rmd documents.

Usage

```
d3po_output(output_id, width = "100%", height = "400px")

render_d3po(expr, env = parent.frame(), quoted = FALSE)

d3po_proxy(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

output_id	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 d3po object
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.
id	Id of plot to create a proxy of.
session	A valid shiny session.

Value

Creates a basic 'htmlwidget' object for 'Shiny' and interactive documents

---

daes *Aesthetics*

Description

Aesthetics of the chart.

Usage

```
daes(x, y, ...)
```

Arguments

x	x-axis mapping.
y	y-axis mapping.
...	Other aesthetic mappings. See the 'Aesthetics' section.

Value

Aesthetics for the plots such as axis (x,y), group, color and/or size

Aesthetics

Valid aesthetics (depending on the geom)

- `x`, `y`: cartesian coordinates.
- `group`: grouping data.
- `subgroup`: subgrouping data (for treemaps).
- `name`: name data.
- `color`: color of geom.
- `size`: size of geom.
- `stack`: `TRUE` or `FALSE` to indicate if the geom should be stacked (for bar charts).
- `tiling`: "squarify" (default), "dice", "slice", "slice-dice" (for treemaps).
- `layout`: "fr", "kk", or any other supported in igraph to set the geom layout (for network charts).
- `gradient`: `TRUE` or `FALSE` to indicate if color should be treated as a gradient palette (for geomaps).
- `sort`: ordering hint for discrete categories. Accepts one of `"asc-x"`, `"desc-x"` (sort by the numeric x/value), or `"asc-y"`, `"desc-y"` (sort by the category/label). Use `"none"` to keep input order.

`national` *National Boundaries Map*

Description

National boundaries for all countries in the 'Natural Earth' repository. The topology has been simplified for better performance in web visualizations and reduced file size.

Usage

```
national
```

Format

An `sf` object with 202 observations and 4 variables.

Variables

- `continent`: Continent name.
- `country`: Country name.
- `country_iso`: ISO 3166-1 alpha-3 country code.
- `geometry`: Simple feature geometry column.

Source

Derived from https://www.naturalearthdata.com/.

Examples

```
national[national$country_iso == "GBR", ]
```

---

po_area *Area*

Description

Plot an area chart.

Usage

```
po_area(d3po, ..., data = NULL, inherit_daes = TRUE)
```

Arguments

d3po	Either the output of d3po() or d3po_proxy().
...	Aesthetics, see daes().
data	Any dataset to use for plot, overrides data passed to d3po().
inherit_daes	Whether to inherit aesthetics previous specified.

Value

an 'htmlwidgets' object with the desired interactive plot

Examples

```
if (interactive()) {
  trade_by_continent <- d3po::trade
  trade_by_continent <- aggregate(
    trade ~ year + reporter_continent,
    data = trade_by_continent,
    FUN = sum
  )

  # Assign colors to continents
  my_pal <- tintin::tintin_pal(option = "Cigars of the Pharaoh")(7)

  names(my_pal) <- c(
    "Africa", "Antarctica", "Asia",
    "Europe", "North America", "Oceania", "South America"
  )

  d3po(trade_by_continent, width = 800, height = 600) |>
```

```
    po_area(daes(
      x = year, y = trade, group = reporter_continent, color = my_pal
    )) |>
    po_labels(
      x = "Year",
      y = "Trade (USD billion)",
      title = "Trade Distribution by Reporter Continent in 2019 and 2023"
    )
}
```

---

po_bar *Bar*

Description

Draw a bar chart.

Usage

```
po_bar(d3po, ..., data = NULL, inherit_daes = TRUE)
```

Arguments

d3po	Either the output of d3po() or d3po_proxy().
...	Aesthetics, see daes().
data	Any dataset to use for plot, overrides data passed to d3po().
inherit_daes	Whether to inherit aesthetics previous specified.

Value

an 'htmlwidgets' object with the desired interactive plot

Examples

```
if (interactive()) {
  trade_by_continent <- d3po::trade
  trade_by_continent <- aggregate(
    trade ~ reporter_continent,
    data = d3po::trade,
    FUN = sum
  )

  # Assign colors to continents
  my_pal <- tintin::tintin_pal()(7)

  names(my_pal) <- c(
    "Africa", "Antarctica", "Asia",
    "Europe", "North America", "Oceania", "South America"
  )
```

```
  d3po(trade_by_continent, width = 800, height = 600) |>
    po_bar(daes(x = reporter_continent, y = trade, color = my_pal)) |>
    po_labels(
      x = "Continent",
      y = "Trade (USD billion)",
      title = "Total Trade by Reporter Continent in 2023"
    )
}
```

---

po_box *Boxplot*

Description

Draw a boxplot.

Usage

```
po_box(d3po, ..., data = NULL, inherit_daes = TRUE)
```

Arguments

d3po	Either the output of d3po() or d3po_proxy().
...	Aesthetics, see daes().
data	Any dataset to use for plot, overrides data passed to d3po().
inherit_daes	Whether to inherit aesthetics previous specified.

Value

an 'htmlwidgets' object with the desired interactive plot

Examples

```
if (interactive()) {
  trade_continent <- d3po::trade
  trade_continent <- aggregate(
    trade ~ reporter_continent + reporter,
    data = trade_continent,
    FUN = sum
  )

  my_pal <- tintin::tintin_pal(option = "Destination Moon")(7)

  names(my_pal) <- c(
    "Africa", "Antarctica", "Asia",
    "Europe", "North America", "Oceania", "South America"
  )
```

```
  d3po(trade_continent, width = 800, height = 600) |>
    po_box(daes(
      x = reporter_continent, y = trade, color = my_pal,
      tooltip = reporter_continent
    )) |>
    po_labels(
      x = "Continent",
      y = "Trade (USD billion)",
      title = "Trade Distribution by Reporter Continent"
    )
}
```

---

po_donut *Donut*

Description

Plot a donut

Usage

```
po_donut(d3po, ..., data = NULL, inherit_daes = TRUE)
```

Arguments

d3po	Either the output of d3po() or d3po_proxy().
...	Aesthetics, see daes().
data	Any dataset to use for plot, overrides data passed to d3po().
inherit_daes	Whether to inherit aesthetics previous specified.

Value

an 'htmlwidgets' object with the desired interactive plot

Examples

```
if (interactive()) {
  trade_by_continent <- d3po::trade[d3po::trade$year == 2023L, ]
  trade_by_continent <- aggregate(
    trade ~ reporter_continent,
    data = d3po::trade,
    FUN = sum
  )

  # Assign colors to continents
  my_pal <- tintin::tintin_pal(option = "The Black Island")(7)

  names(my_pal) <- c(
    "Africa", "Antarctica", "Asia",
```

```
    "Europe", "North America", "Oceania", "South America"
  )

  trade_by_continent$color <- my_pal[trade_by_continent$reporter_continent]

  d3po(trade_by_continent, width = 800, height = 600) |>
   po_donut(daes(size = trade, group = reporter_continent, inner_radius = 0.3, color = color)) |>
    po_labels(title = "Trade Share by Reporter Continent in 2023")
}
```

`po_download` *Download*

Description

Show/hide the download button.

Usage

```
po_download(d3po, show = TRUE)
```

Arguments

`d3po`	A 'd3po' or 'd3proxy' object.
`show`	Logical indicating whether to show (TRUE) or hide (FALSE) the download button.

Value

Appends download button settings to an 'htmlwidgets' object

`po_font` *Font*

Description

Edit the font used in a chart.

Usage

```
po_font(d3po, family = "Fira Sans", size = 16, transform = "none")
```

Arguments

`d3po`	Either the output of `d3po()` or `d3po_proxy()`.
`family`	family font to use ("Roboto", "Merriweather", etc.).
`size`	size to use (10, 11, 12, etc. overrides auto-sizing).
`transform`	transformation to use for the title ("lowercase", "uppercase", "capitalize", "none").

Value

Appends custom font to an 'htmlwidgets' object

---

po_format *Format*

Description

Precompute formatted label columns from expressions and attach them to the widget data. Accepts named expressions like `x = round(varx, 2)` or `y = format(varY, big.mark = ",")`. The formatted columns are added to `d3po$x$data` with names `__label_<name>` and registered in `d3po$x$formatted_cols` for the renderer to use.

Usage

```
po_format(d3po, ...)
```

Arguments

`d3po`	Either the output of `d3po()` or `d3po_proxy()`.
`...`	Named formatting expressions (as unevaluated calls). Each name should correspond to an aesthetic (e.g. `x`, `y`, `tooltip`, etc.).

---

po_geomap *Geomap*

Description

Plot a geomap using sf spatial objects

Usage

```
po_geomap(d3po, ..., data = NULL, inherit_daes = TRUE, limits = NULL)
```

Arguments

`d3po`	Either the output of `d3po()` or `d3po_proxy()`.
`...`	Aesthetics, see `daes()`.
`data`	Any dataset to use for plot, overrides data passed to `d3po()`.
`inherit_daes`	Whether to inherit aesthetics previous specified.
`limits`	A numeric vector of length 2 specifying the minimum and maximum values for the color scale.

Value

an 'htmlwidgets' object with the desired interactive plot

Examples

```
if (interactive()) {
  world <- d3po::national

  # Fix geometries that cross the antimeridian (date line) to avoid horizontal lines
  # This affects Russia, Fiji, and other countries spanning the 180° meridian
  world$geometry <- sf::st_wrap_dateline(world$geometry, options = c("WRAPDATELINE=YES"))

  total_trade <- d3po::trade[
    d3po::trade$year == 2023L,
    c("reporter", "reporter_continent", "trade")
  ]
  total_trade <- aggregate(trade ~ reporter, data = total_trade, FUN = sum)
  colnames(total_trade) <- c("country", "trade")

  world <- merge(
    world,
    total_trade,
    by = "country",
    all.x = TRUE,
    all.y = FALSE
  )

  my_pal <- tintin::tintin_pal(option = "The Calculus Affair")(7)

  names(my_pal) <- c(
    "Africa", "Antarctica", "Asia",
    "Europe", "North America", "Oceania", "South America"
  )

  d3po(world, width = 800, height = 600) |>
    po_geomap(daes(group = country, size = trade, color = my_pal, tooltip = country)) |>
    po_labels(title = "Trade Volume by Country in 2023")
}
```

---

po_labels *Labels*

Description

Edit labels positioning in a treemap.

Usage

```
po_labels(
  d3po,
  x = NULL,
  y = NULL,
  title = NULL,
  subtitle = NULL,
  labels = NULL,
  align = "left-top"
)
```

Arguments

d3po	Either the output of d3po() or d3po_proxy().
x	Optional x-axis label.
y	Optional y-axis label.
title	Optional title for the chart.
subtitle	Optional subtitle for the chart.
labels	Optional character vector or JavaScript function for custom label fields for treemaps.
align	Label alignment for treemaps. Must be one of "left-top", "center-middle", or "right-top".

Value

Appends custom labels to an 'htmlwidgets' object

---

po_line *Line*

Description

Plot an line chart.

Usage

```
po_line(d3po, ..., data = NULL, inherit_daes = TRUE)
```

Arguments

d3po	Either the output of d3po() or d3po_proxy().
...	Aesthetics, see daes().
data	Any dataset to use for plot, overrides data passed to d3po().
inherit_daes	Whether to inherit aesthetics previous specified.

Value

an 'htmlwidgets' object with the desired interactive plot

Examples

```
if (interactive()) {
  trade_by_continent <- d3po::trade
  trade_by_continent <- aggregate(
    trade ~ year + reporter_continent,
    data = trade_by_continent,
    FUN = sum
  )

  # Assign colors to continents
  my_pal <- tintin::tintin_pal(option = "The Broken Ear")(7)

  names(my_pal) <- c(
    "Africa", "Antarctica", "Asia",
    "Europe", "North America", "Oceania", "South America"
  )

  d3po(trade_by_continent, width = 800, height = 600) |>
    po_line(daes(x = year, y = trade, group = reporter_continent, color = my_pal)) |>
    po_labels(
      x = "Year",
      y = "Trade (USD billion)",
      title = "Trade Distribution by Reporter Continent in 2019 and 2023"
    )
}
```

---

po_network *Network*

Description

Draw a network graph showing relationships between entities. Requires an igraph object with nodes (vertices) and links (edges). Node size can represent counts or other metrics.

Usage

```
po_network(d3po, ..., data = NULL, inherit_daes = TRUE)
```

Arguments

d3po	Either the output of d3po() or d3po_proxy().
...	Aesthetics, see daes().
data	Any dataset to use for plot, overrides data passed to d3po().
inherit_daes	Whether to inherit aesthetics previous specified.

Value

Appends nodes arguments to a network-specific 'htmlwidgets' object

Examples

```
if (interactive()) {
  trade_network <- d3po::trade[d3po::trade$year == 2023L, ]
  trade_network <- aggregate(
    trade ~
      reporter_iso + partner_iso + reporter_continent + partner_continent,
    data = trade_network, FUN = sum
  )

  # subset to 10 largest connection per reporter country
  trade_network <- do.call(
    rbind,
    lapply(
      split(trade_network, trade_network$reporter_iso),
      function(df) head(df[order(-df$trade), ], 10)
    )
  )

  # Create vertex (node) attributes for coloring and sizing
  # Get unique countries with their continents and trade volumes
  vertices <- unique(rbind(
    data.frame(
      name = trade_network$reporter_iso,
      continent = trade_network$reporter_continent,
      stringsAsFactors = FALSE
    ),
    data.frame(
      name = trade_network$partner_iso,
      continent = trade_network$partner_continent,
      stringsAsFactors = FALSE
    )
  ))

  # Remove duplicates
  vertices <- vertices[!duplicated(vertices$name), ]

  # Calculate total trade volume per country (as reporter)
  trade_volume <- aggregate(trade ~ reporter_iso, data = trade_network, FUN = sum)
  colnames(trade_volume) <- c("name", "trade_volume")

  # Merge trade volume with vertices
  vertices <- merge(vertices, trade_volume, by = "name", all.x = TRUE)
  vertices$trade_volume[is.na(vertices$trade_volume)] <- 0

  # Assign colors to continents
  my_pal <- tintin::tintin_pal(option = "The Blue Lotus")(7)

  names(my_pal) <- c(
```

```
    "Africa", "Antarctica", "Asia",
    "Europe", "North America", "Oceania", "South America"
  )

  # Add color column based on continent
  vertices$color <- my_pal[vertices$continent]

  # Create igraph object with vertex attributes
  g <- graph_from_data_frame(trade_network, directed = TRUE, vertices = vertices)

  # Create the network visualization
  d3po(g, width = 800, height = 600) |>
    po_network(daes(size = trade_volume, color = color, layout = "fr")) |>
    po_labels(title = "Trade Network by Country in 2023")
}
```

`po_pie` *Pie*

Description

Plot a pie

Usage

```
po_pie(d3po, ..., data = NULL, inherit_daes = TRUE)
```

Arguments

`d3po`	Either the output of `d3po()` or `d3po_proxy()`.
`...`	Aesthetics, see `daes()`.
`data`	Any dataset to use for plot, overrides data passed to `d3po()`.
`inherit_daes`	Whether to inherit aesthetics previous specified.

Value

an 'htmlwidgets' object with the desired interactive plot

Examples

```
if (interactive()) {
  trade_by_continent <- d3po::trade[d3po::trade$year == 2023L, ]
  trade_by_continent <- aggregate(
    trade ~ reporter_continent,
    data = d3po::trade,
    FUN = sum
  )
```

```
  # Assign colors to continents
  my_pal <- tintin::tintin_pal(option = "The Black Island")(7)

  names(my_pal) <- c(
    "Africa", "Antarctica", "Asia",
    "Europe", "North America", "Oceania", "South America"
  )

  d3po(trade_by_continent, width = 800, height = 600) |>
    po_pie(daes(size = trade, group = reporter_continent, color = my_pal)) |>
    po_labels(title = "Trade Share by Reporter Continent in 2023")
}
```

`po_scatter` *Scatter*

Description

Plot an scatter chart.

Usage

```
po_scatter(d3po, ..., data = NULL, inherit_daes = TRUE)
```

Arguments

`d3po`	Either the output of `d3po()` or `d3po_proxy()`.
`...`	Aesthetics, see `daes()`.
`data`	Any dataset to use for plot, overrides data passed to `d3po()`.
`inherit_daes`	Whether to inherit aesthetics previous specified.

Value

an 'htmlwidgets' object with the desired interactive plot

Examples

```
if (interactive()) {
  # Create a wide dataset with x = 2019 and y = 2023 trade values
  trade_wide_2019 <- d3po::trade[d3po::trade$year == 2019L, c("reporter", "trade")]
  trade_wide_2019 <- aggregate(trade ~ reporter, data = trade_wide_2019, FUN = sum)

  trade_wide_2023 <- d3po::trade[d3po::trade$year == 2023L, c("reporter", "trade")]
  trade_wide_2023 <- aggregate(trade ~ reporter, data = trade_wide_2023, FUN = sum)

  trade_wide <- merge(
    trade_wide_2019,
    trade_wide_2023,
    by = "reporter",
```

```
    suffixes = c("_2019", "_2023")
  )

  my_pal <- tintin::tintin_pal(option = "red_rackhams_treasure")(7)

  d3po(trade_wide, width = 800, height = 600) |>
    po_scatter(daes(x = trade_2019, y = trade_2023, group = reporter, color = my_pal)) |>
    po_labels(
      x = "Trade in 2019 (USD billion)",
      y = "Trade in 2023 (USD billion)",
      title = "Trade Volume by Country in 2019 and 2023"
    )
}
```

`po_theme` *Theme*

Description

Manually set colors used by D3po for axis/axis-labels/title, tooltips/download menu background, and chart background. This allows you to override page themes (Tabler/Shiny) and force D3po to render with readable contrast.

Usage

```
po_theme(d3po, axis = NULL, tooltips = NULL, background = NULL)
```

Arguments

`d3po`	Either the output of `d3po()` or `d3po_proxy()`.
`axis`	Hex color string for axis lines and axis/label/title fill (e.g. "#fff").
`tooltips`	Hex color string for tooltip / download menu background (e.g. "#000").
`background`	Hex color string for chart background (e.g. "#fff").

Value

Appends theme settings to an 'htmlwidgets' object

po_tooltip *Tooltip*

Description

Set a custom tooltip template for a chart. The template can be a literal string (e.g. `<b>{name}</b><br>Value: {value}`) which will be evaluated server-side by substituting column values.

Usage

```
po_tooltip(d3po, template)
```

Arguments

d3po	Either the output of d3po() or d3po_proxy().
template	A character string template or formatter expression.

Value

Appends tooltip settings to an 'htmlwidgets' object

po_treemap *Treemap*

Description

Plot a treemap

Usage

```
po_treemap(d3po, ..., data = NULL, inherit_daes = TRUE)
```

Arguments

d3po	Either the output of d3po() or d3po_proxy().
...	Aesthetics, see daes().
data	Any dataset to use for plot, overrides data passed to d3po().
inherit_daes	Whether to inherit aesthetics previous specified.

Value

an 'htmlwidgets' object with the desired interactive plot

Examples

```
if (interactive()) {
  trade_by_continent <- d3po::trade[d3po::trade$year == 2023L, ]
 trade_by_continent <- aggregate(trade ~ reporter_continent, data = trade_by_continent, FUN = sum)

  my_pal <- tintin::tintin_pal(option = "The Secret of the Unicorn")(7)

  names(my_pal) <- c(
    "Africa", "Antarctica", "Asia",
    "Europe", "North America", "Oceania", "South America"
  )

  d3po(trade_by_continent, width = 800, height = 600) |>
    po_treemap(daes(
      size = trade, group = reporter_continent,
      color = my_pal, tiling = "Squarify"
    )) |>
    po_labels(title = "Trade Share by Continent in 2023")
}
```

---

subnational *Subnational Boundaries Map*

Description

Subnational boundaries for the available countries in the 'Natural Earth' repository. The topology has been simplified for better performance in web visualizations and reduced file size. Subnational regions include states, provinces, and equivalent administrative divisions.

Usage

```
subnational
```

Format

An `sf` object with 4,596 observations and 6 variables.

Variables

- `continent`: Continent name.
- `country`: Country name.
- `region`: Subnational region name.
- `country_iso`: ISO 3166-1 alpha-3 country code.
- `region_iso`: ISO 3166-2 subnational region code.
- `geometry`: Simple feature geometry column.

Source

Derived from https://www.naturalearthdata.com/.

Examples

```
subnational[subnational$country_iso == "GBR", ]
```

---

trade *World Trade 2019 and 2023*

Description

Bilateral trade flows between countries for the years 2019 and 2023. Trade values are expressed in USD billion dollars.

Usage

```
trade
```

Format

A data frame with 5,223 observations and 8 variables.

Variables

- `year`: Year of the trade data (2019 or 2023).
- `reporter_continent`: Continent of the reporting country.
- `partner_continent`: Continent of the partner country.
- `reporter`: Name of the reporting country.
- `partner`: Name of the partner country.
- `reporter_iso`: ISO 3166-1 alpha-3 code of the reporting country.
- `partner_iso`: ISO 3166-1 alpha-3 code of the partner country.
- `trade`: Trade value in USD billion dollars. Reporter (importer-side) figures are used.

Source

Derived from the UN Comtrade database (https://comtrade.un.org/).

Examples

```
head(trade[trade$year == 2023L & trade$reporter_iso == "gbr", ])
```

Index

∗ **datasets**
national, 4
subnational, 19
trade, 20

d3po(), *5–10*, *12, 13*, *15–18*
d3po-exports, 2
d3po-shiny, 2
d3po_output *(*d3po-shiny*)*, 2
d3po_proxy *(*d3po-shiny*)*, 2
d3po_proxy(), *5–10*, *12, 13*, *15–18*
daes, 3
daes(), *5–8*, *10*, *12, 13*, *15, 16*, *18*

JS *(*d3po-exports*)*, 2

national, 4

po_area, 5
po_bar, 6
po_box, 7
po_donut, 8
po_download, 9
po_font, 9
po_format, 10
po_geomap, 10
po_labels, 11
po_line, 12
po_network, 13
po_pie, 15
po_scatter, 16
po_theme, 17
po_tooltip, 18
po_treemap, 18

render_d3po *(*d3po-shiny*)*, 2

subnational, 19

trade, 20