R topics documented:

d3po .............................................................. 2
d3po-exports .................................................. 3
d3po-shiny ...................................................... 3
d3po_template .................................................. 4
daes ............................................................... 4
maps ............................................................... 5
map_ids ......................................................... 5
pokemon ......................................................... 6
pokemon_network .............................................. 7
po_area ........................................................ 7
po_background ............................................... 8
po_bar .......................................................... 9
po_box .......................................................... 10
po_donut ........................................................ 10
po_font ........................................................ 11
po_geemap ..................................................... 12
po_labels ....................................................... 13
po_legend ...................................................... 13
po_line ........................................................ 14
po_network ..................................................... 15
po_pie .......................................................... 15
po_scatter ..................................................... 16
po_title ......................................................... 18
po_treemap .................................................... 18

Index 20

d3po An htmlwidget interface to the d3po javascript chart library

Description

This function provides 'd3po' methods from R console

Usage

d3po(data = NULL, ..., width = NULL, height = NULL, elementId = NULL)

Arguments

data d3po need explicit specified data objects formatted as JSON, and this parameter passed it from R.

... Aesthetics to pass, see daes()

width Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
Description

**d3po-exports**

D3po (re)exported methods

**d3po-shiny**

Shiny bindings for 'd3po'

Usage

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

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

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output_id</td>
<td>output variable to read from</td>
</tr>
<tr>
<td>width, height</td>
<td>Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.</td>
</tr>
<tr>
<td>expr</td>
<td>An expression that generates a d3po object</td>
</tr>
<tr>
<td>env</td>
<td>The environment in which to evaluate expr.</td>
</tr>
<tr>
<td>quoted</td>
<td>Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.</td>
</tr>
<tr>
<td>id</td>
<td>Id of plot to create a proxy of.</td>
</tr>
<tr>
<td>session</td>
<td>A valid shiny session.</td>
</tr>
</tbody>
</table>
**Value**

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

---

**d3po_template**

*Create a new d3po templated project*

---

**Description**

Create a new d3po templated project

**Usage**

```r
d3po_template(path)
```

**Arguments**

- `path`: The path to create the new project in

---

**daes**

*Aesthetics*

---

**Description**

Aesthetics of the chart.

**Usage**

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

**Arguments**

- `x, y, ...`: List of name value pairs giving aesthetics to map to variables. The names for `x` and `y` aesthetics are typically omitted because they are so common; all other aspects must be named.

**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.
- color: color of geom.
- size: size of geom.
- layout: layout of geom (nicely, fr, kk, graphopt, drl, lgl, mds, sugiyama), in quotes.
Description

World, continent and country maps. These maps are provided as R lists structured by following the 'topojson' standard. The maps are organized in sub-lists by continent and here I provide maps for both the continents and the countries. There are missing states or regions because those could not be found in the original maps.

Usage

maps

Format

A list object with 6 elements (one per continent). The Americas are separated in North America and South America.

Details

Missing in Asia: 'Siachen Glacier (JK)', 'Scarborough Reef (SH)', and 'Spratly Islands (SP)'.
Missing in Europe: 'Vatican City (VA)'.
Missing in North America: 'Bajo Nuevo Bank (BU)', 'Serranilla Bank (SW)', and 'United States Minor Outlying Islands (UM)'.
Missing in Oceania: 'Federated States of Micronesia (FM)', 'Marshall Islands (MH)', and 'Tuvalu (TV)'.
Consider all these maps as referential and unofficial.

Source

Adapted from Natural Earth.

map_ids

Extract the IDs from a Map

Description

Extract the IDs from a Map

Usage

map_ids(map)
Arguments

map A map object

Value

A tibble containing IDs and names

Examples

map <- map_ids(maps$south_america$continent)

---

pokemon

Description

Statistical information about 151 Pokemon from Nintendo RPG series.

Usage

pokemon

Format

A data frame with 151 observations and 15 variables.

Variables

- id: Pokedex number.
- name: Pokedex name.
- height: Height in meters.
- weight: Weight in kilograms.
- base_experience: How much the Pokemon has battled.
- type_1: Primary Pokemon type (i.e. Grass, Fire and Water)
- type_2: Secondary Pokemon type (i.e. Poison, Dragon and Ice)
- attack: How much damage a Pokemon deals when using a technique.
- defense: How much damage a Pokemon receives when it is hit by a technique.
- hp: How much damage a Pokemon can receive before fainting.
- special_attack: How much damage a Pokemon deals when using a special technique.
- specialDefense: How much damage a Pokemon receives when it is hit by a special technique.
- speed: Determines the order of Pokemon that can act in battle, if the speed is tied then the 1st move is assigned at random.
- color_1: Hex color code for Type 1.
- color_2: Hex color code for Type 2.
pokemon_network

Source
Adapted from highcharter package.

pokemon_network

Description
Connections between Pokemon types based on Type 1 and 2.

Usage
pokemon_network

Format
A igraph object with 17 vertices (nodes) and 26 edges (arcs).

Source
Adapted from the highcharter package.

po_area

Area

Description
Plot an area chart.

Usage
po_area(d3po, ..., data = NULL, inherit_daes = TRUE, stack = FALSE)

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.
stack Whether to stack the series.

Value
an 'htmlwidgets' object with the desired interactive plot
Examples

```r
# library(dplyr)
# dout <- pokemon %>%
# filter(
#   type_1 == "water"
# ) %>%
# group_by(type_1, color_1) %>%
# reframe(
#   probability = c(0, 0.25, 0.5, 0.75, 1),
#   quantile = quantile(speed, probability)
# )

dout <- data.frame(
  type_1 = rep("water", 5),
  color_1 = rep("#6890F0", 5),
  probability = c(0, 0.25, 0.5, 0.75, 1),
  quantile = c(15, 57.25, 70, 82, 115)
)

d3po(dout) %>%
  po_area(daes(
    x = probability, y = quantile, group = type_1,
    color = color_1
  )) %>%
  po_title("Sample Quantiles for Water Pokemon Speed")
```

---

**po_background**

<table>
<thead>
<tr>
<th><strong>Background</strong></th>
</tr>
</thead>
</table>

**Description**

Add a background to a chart.

**Usage**

```r
po_background(d3po, background = "#fff")
```

**Arguments**

- `d3po`: Either the output of `d3po()` or `d3po_proxy()`.
- `background`: Background to add (hex code).

**Value**

Appends custom background to an 'htmlwidgets' object.
Description

Draw a bar chart.

Usage

\texttt{po\_bar(d3po, \ldots, data = NULL, inherit\_daes = TRUE)}

Arguments

- \texttt{d3po}: Either the output of \texttt{d3po()} or \texttt{d3po\_proxy()}.
- \ldots: Aesthetics, see \texttt{daes()}.
- \texttt{data}: Any dataset to use for plot, overrides data passed to \texttt{d3po()}.
- \texttt{inherit\_daes}: Whether to inherit aesthetics previous specified.

Value

An \texttt{htmlwidgets} object with the desired interactive plot.

Examples

```
# library(dplyr)
# dout <- pokemon %>%
#   group_by(type_1, color_1) %>%
#   count()

dout <- data.frame(
  type_1 = c(
    "bug", "dragon", "electric", "fairy", "fighting",
    "fire", "ghost", "grass", "ground", "ice",
    "normal", "poison", "psychic", "rock", "water" ),
  color_1 = c(
    "#A8B820", "#7038F8", "#F8D030", "#EE99AC", "#C03028",
    "#F08030", "#78C850", "#EE99AC", "#80D080",
    "#A8A878", "#A8A878", "#80D080", "#80D080", "#80D080"
  ),
  n = c(
    12, 3, 9, 2, 7,
    12, 3, 12, 8, 2,
    22, 14, 8, 9, 28
  )
)

d3po(dout) %>%
  po_bar(daes(x = type_1, y = n, color = color_1)) %>%
  po_title("Share of Pokemon by main type")
```
po_box  

**Boxplot**

**Description**

Draw a boxplot.

**Usage**

```r
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**

```r
d3po(pokemon) %>%
  po_box(daes(x = type_1, y = speed, color = color_1)) %>%
  po_title("Distribution of Pokemon speed by main type")
```

---

po_donut  

**Donut**

**Description**

Plot a donut

**Usage**

```r
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

```r
# library(dplyr)
# dout <- pokemon %>%
#   group_by(type_1, color_1) %>%
#   count()

dout <- data.frame(
  type_1 = c(
    "bug", "dragon", "electric", "fairy", "fighting",
    "fire", "ghost", "grass", "ground", "ice",
    "normal", "poison", "psychic", "rock", "water"
  ),
  color_1 = c(
    "#A8B820", "#7038F8", "#F8D030", "#EE99AC", "#C03028",
    "#F08030", "#78C850", "#E0C068", "#98D8D8",
    "#A8A820", "#A040A0", "#F85888", "#88A038", "#5890F0"
  ),
  n = c(
    12, 3, 9, 2, 7,
    12, 3, 12, 8, 2,
    22, 14, 8, 9, 28
  )
)

d3po(dout) %>%
  po_donut(daes(size = n, group = type_1, color = color_1)) %>%
  po_title("Share of Pokemon by main type")
```

<table>
<thead>
<tr>
<th>po_font</th>
<th>Font</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

Edit the font used in a chart.

Usage

```r
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").
**po_geomap**

**Value**

Appends custom font to an 'htmlwidgets' object

---

**po_geomap**  
**Geomap**

**Description**

Plot a geomap

**Usage**

```r
po_geomap(d3po, ..., data = NULL, map = 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()`.
- `map`: Map to use (i.e., any valid list or topojson file such as `maps$south_america` or `jsonlite::fromJSON("south_america.topojson", simplifyVector = F)`).
- `inherit_daes`: Whether to inherit aesthetics previous specified.

**Value**

an 'htmlwidgets' object with the desired interactive plot

**Examples**

```r
dout <- map_ids(d3po::maps$asia$japan)
dout$value <- ifelse(dout$id == "TK", 1L, NA)
dout$color <- ifelse(dout$id == "TK", "#bd0029", NA)

d3po(dout) %>%
  po_geomap(
    daes(
      group = id, color = color, size = value,
      tooltip = name
    ),
    map = d3po::maps$asia$japan
  ) %>%
  po_title("Pokemon was created in the Japanese city of Tokyo")
```
**po_labels**  

*Labels*

**Description**
Edit labels positioning in a chart.

**Usage**
```r
po_labels(d3po, align = "center", valign = "middle", resize = TRUE)
```

**Arguments**
- `d3po` Either the output of `d3po()` or `d3po_proxy()`.
- `align` horizontal alignment ("left", "center", "right", "start", "middle", "end").
- `valign` vertical alignment ("top", "middle", "bottom").
- `resize` resize labels text (TRUE or FALSE).

**Value**
Appends custom labels to an `htmlwidgets` object

---

**po_legend**  

*Legend*

**Description**
Add a legend to a chart.

**Usage**
```r
po_legend(d3po, legend)
```

**Arguments**
- `d3po` Either the output of `d3po()` or `d3po_proxy()`.
- `legend` legend to add.

**Value**
Appends custom legend to an `htmlwidgets` object
Description

Plot an line chart.

Usage

\[
\text{po_line}(d3po, ..., \text{data} = \text{NULL}, \text{inherit_daes} = \text{TRUE})
\]

Arguments

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

Value

an 'htmlwidgets' object with the desired interactive plot

Examples

```r
# library(dplyr)
# dout <- pokemon %>%
# filter(
#  type_1 == "water"
# ) %>%
# group_by(type_1, color_1) %>%
# reframe(
#  probability = c(0, 0.25, 0.5, 0.75, 1),
#  quantile = quantile(speed, probability)
# )

dout <- data.frame(
  type_1 = rep("water", 5),
  color_1 = rep("#6890F0", 5),
  probability = c(0, 0.25, 0.5, 0.75, 1),
  quantile = c(15, 57.25, 70, 82, 115)
)

d3po(dout) %>%
  po_line(daes(
    x = probability, y = quantile, group = type_1,
    color = color_1
  )) %>%
  po_title("Sample Quantiles for Water Pokemon Speed")
```
#### po_network

**Description**

Draw a network.

**Usage**

```r
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**

```r
d3po(pokemon_network) %>%
  po_network(daes(size = size, color = color, layout = "kk")) %>%
  po_title("Connections Between Pokemon Types")
```

#### po_pie

**Description**

Plot a pie

**Usage**

```r
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

```
# library(dplyr)
# dout <- pokemon %>%
#   group_by(type_1, color_1) %>%
#   count()

dout <- data.frame(
  type_1 = c(
    "bug", "dragon", "electric", "fairy", "fighting",
    "fire", "ghost", "grass", "ground", "ice",
    "normal", "poison", "psychic", "rock", "water"
  ),
  color_1 = c(
    "#A8B820", "#7038F8", "#F8D030", "#EE99AC", "#C03028",
    "#F08030", "#705898", "#78C850", "#E0C068", "#98D8D8",
    "#A8A828", "#A040A0", "#F85888", "#B8A038", "#6890F0"
  ),
  n = c(
    12, 3, 9, 2, 7,
    12, 3, 12, 8, 2,
    22, 14, 8, 9, 28
  )
)
d3po(dout) %>%
  po_pie(daes(size = n, group = type_1, color = color_1)) %>%
  po_title("Share of Pokemon by main type")
```

---

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

```r
# library(dplyr)
# dout <- pokemon %>%
# group_by(type_1, color_1) %>%
# summarise(
#   attack = mean(attack),
#   defense = mean(defense)
# ) %>%
# mutate(log_attack_x_defense = log(attack * defense))

# dout <- data.frame(
#   type_1 = c(
#     "bug", "dragon", "electric", "fairy", "fighting",
#     "fire", "ghost", "grass", "ground", "ice",
#     "normal", "poison", "psychic", "rock", "water"
#   ),
#   color_1 = c(
#     "#A8B820", "#F8D030", "#EE99AC", "#C03028",
#     "#F08030", "#705898", "#78C850", "#E0C068", "#98D8D8",
#     "#A8A878", "#A040A0", "#F85888", "#B8A038", "#6890F0"
#   ),
#   attack = c(
#     63.7, 94, 62, 57.5, 102.8,
#     83.9, 50, 70.6, 81.8, 67.5,
#     67.7, 74.4, 60.1, 82.2, 70.2
#   ),
#   defense = c(
#     57, 68.3, 64.6, 60.5, 61,
#     62.5, 45, 69.5, 86.2, 67.5,
#     53.5, 67, 57.5, 110, 77.5
#   ),
#   log_attack_x_defense = c(
#     8.1, 8.7, 8.2, 8.1, 8.7,
#     8.5, 7.7, 8.5, 8.8, 8.4,
#     8.1, 8.5, 8.1, 9.1, 8.6
#   )
# )

d3po(dout) %>%
  po_scatter(daes(
    x = defense, y = attack,
    size = log_attack_x_defense, group = type_1, color = color_1
  )) %>%
  po_title("Pokemon Mean Attack vs Mean Defense by Main Type")
```
### po_title

**Description**
Add a title to a chart.

**Usage**
```r
po_title(d3po, title)
```

**Arguments**
- `d3po`: Either the output of `d3po()` or `d3po_proxy()`.
- `title`: Title to add.

**Value**
Appends a title to an 'htmlwidgets' object

### po_treemap

**Description**
Plot a treemap

**Usage**
```r
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

```r
# library(dplyr)
# dout <- pokemon %>%
#  group_by(type_1, color_1) %>%
#  count()

dout <- data.frame(
  type_1 = c(
    "bug", "dragon", "electric", "fairy", "fighting",
    "fire", "ghost", "grass", "ground", "ice",
    "normal", "poison", "psychic", "rock", "water"
  ),
  color_1 = c(
    "#A8B820", "#7038F8", "#F8D030", "#EE99AC", "#C03028",
    "#F08030", "#705898", "#78C850", "#E0C068", "#98D8D8",
    "#A8A878", "#A040A0", "#F85888", "#B8A038", "#6890F0"
  ),
  n = c(
    12, 3, 9, 2, 7,
    12, 3, 12, 8, 2,
    22, 14, 8, 9, 28
  )
)

d3po(dout) %>
  po_treemap(daes(size = n, group = type_1, color = color_1)) %>
  po_title("Share of Pokemon by main type")
```
Index

* datasets
  - maps, 5
  - pokemon, 6
  - pokemon_network, 7
  - %>% (d3po-exports), 3
  - d3po, 2
  - d3po(), 7–16, 18
  - d3po-exports, 3
  - d3po-shiny, 3
  - d3po_output (d3po-shiny), 3
  - d3po_proxy (d3po-shiny), 3
  - d3po_proxy(), 7–16, 18
  - d3po_template, 4
  - daes, 4
  - daes(), 2, 7, 9, 10, 12, 14–16, 18
  - JS (d3po-exports), 3
  - map_ids, 5
  - maps, 5
  - po_area, 7
  - po_background, 8
  - po_bar, 9
  - po_box, 10
  - po_donut, 10
  - po_font, 11
  - po_geomap, 12
  - po_labels, 13
  - po_legend, 13
  - po_line, 14
  - po_network, 15
  - po_pie, 15
  - po_scatter, 16
  - po_title, 18
  - po_treemap, 18
  - pokemon, 6
  - pokemon_network, 7
  - render_d3po (d3po-shiny), 3