Package 'd3r'

June 27, 2017

Type Package

Title 'd3.js' Utilities for R

Version 0.6.6

Date 2017-06-26

Maintainer Kent Russell <kent.russell@timelyportfolio.com>

URL https://github.com/timelyportfolio/d3r

BugReports https://github.com/timelyportfolio/d3r/issues

Description Helper functions for using 'd3.js' in R.

License BSD_3_clause + file LICENSE

Encoding UTF-8

Imports dplyr, htmltools, tidyr

Suggests jsonlite, listviewer, purrr, testthat

Enhances igraph, partykit, rpart, treemap, V8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Author Mike Bostock [aut, cph] (d3.js library in htmlwidgets/lib,
http://d3js.org),
Kent Russell [aut, cre] (R interface)

Repository CRAN

Date/Publication 2017-06-27 06:37:53 UTC

R topics documented:

`change_to_name` *Change Column Name in Children to "name"*

Description

Change Column Name in Children to "name"

Usage

```
change_to_name(x, column = 1)
```

Arguments

`x`	`data.frame` or `data.frame` derivative, such as `tibble`
`column`	column to convert

Value

`data.frame`

`d3_dep_v3` *'d3.js' Dependency for Version 3*

Description

'd3.js' Dependency for Version 3

Usage

```
d3_dep_v3(offline = TRUE)
```

Arguments

`offline`	`logical` to specify whether to use a local copy of d3.js (`TRUE`) or use cdn (`FALSE`)

Value

htmltools::htmlDependency

Examples

```
library(d3r)
library(htmltools)

attachDependencies(tagList(),d3_dep_v3())
```

---

d3_dep_v4 *'d3.js' Dependency for Version 4*

Description

'd3.js' Dependency for Version 4

Usage

```
d3_dep_v4(offline = TRUE)
```

Arguments

offline	logical to specify whether to use a local copy of d3.js (TRUE) or use cdn (FALSE)

Value

htmltools::htmlDependency

Examples

```
library(d3r)
library(htmltools)

attachDependencies(tagList(),d3_dep_v4())
```

d3_igraph *Convert 'igraph' to 'd3.js'*

Description

Convert 'igraph' to 'd3.js'

Usage

```
d3_igraph(igrf = NULL, json = TRUE)
```

Arguments

igrf	igraph
json	logical to return as JSON

Value

list

Examples

```
## Not run:
library(igraph)
library(igraphdata)
library(d3r)

# with random graph
d3r::d3_igraph(igraph::sample_pa(100))

# check case where vertices 0 cols
d3_igraph(igraph::watts.strogatz.game(1, 50, 4, 0.05))

# with karate from igraphdata
#  notice graph attributes are added
data("karate",package="igraphdata")
(karate_d3 <- d3r::d3_igraph(karate))

listviewer::jsonedit(karate_d3)

data("kite",package="igraphdata")
listviewer::jsonedit(d3_igraph(kite))

## End(Not run)
```

d3_json *Create 'JSON' that 'd3.js' Expects*

Description

Create 'JSON' that 'd3.js' Expects

Usage

```
d3_json(x = NULL, strip = TRUE)
```

Arguments

x	data, usually in the form of data.frame or list
strip	logical to remove outer array. Use strip=TRUE for hierarchies from d3_nest

Value

string of 'JSON' data

d3_nest *Convert a* data.frame *to a 'd3.js' Hierarchy*

Description

Convert a data.frame to a 'd3.js' Hierarchy

Usage

```
d3_nest(data = NULL, value_cols = character(), root = "root",
  json = TRUE)
```

Arguments

data	data.frame or data.frame derivative, such as tibble
value_cols	character vector with the names of the columns to use as data
root	character name of the root level of the hierarchy
json	logical to return as JSON

Value

nested data.frame

Examples

```
# convert Titanic to a nested d3 hierarchy

# devtools::install_github("timelyportfolio/d3r")
library(d3r)
library(dplyr)

titanic_df <- data.frame(Titanic)
tit_tb <- titanic_df %>%
  select(Class,Age,Survived,Sex,Freq) %>%
  d3_nest(value_cols="Freq", root="titanic")

tit_tb

# see as tibble
titanic_df %>%
  select(Class,Age,Survived,Sex,Freq) %>%
  d3_nest(value_cols="Freq", root="titanic", json=FALSE)

# see the structure with listviewer
tit_tb %>%
  listviewer::jsonedit()
## Not run:
  library(treemap)
  library(d3r)
  library(dplyr)
  library(tidyr)

  treemap::random.hierarchical.data() %>%
    d3_nest(value_cols = "x")

  # use example from ?treemap
  data(GNI2014)
  treemap(
    GNI2014,
    index=c("continent", "iso3"),
    vSize="population",
    vColor="GNI",
    type="value",
    draw=FALSE
  ) %>%
    {.$tm} %>%
    select(continent,iso3,color,vSize) %>%
    d3_nest(value_cols = c("color", "vSize"))

## End(Not run)
```

---

d3_party *Convert partykit to d3.js hierarchy*

Description

This thing is not even close to being done, so please help with ideas and contributions.

Usage

```
d3_party(tree = NULL, json = TRUE)
```

Arguments

tree	partykit object to be converted
json	logical to return list or json

Value

list or json depending on json arg

Examples

```
## Not run:

  library(d3r)
  # from ?rpart
  data("kyphosis", package="rpart")
  d3_party(
    rpart::rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
  )

  # if you want the list instead of json
  d3_party(
    rpart::rpart(Kyphosis ~ Age + Number + Start, data = kyphosis),
    json = FALSE
  )

  # with ctree instead of rpart
  #  using example from ?ctree
  d3_party(partykit::ctree(Species ~ .,data = iris))


  #devtools::install_github("timelyportfolio/d3treeR")

  library(d3treeR)

  d3tree2(
    d3_party(
      rpart::rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
    ),
    celltext = "rule",
    valueField = "n"
  )


## End(Not run)
```

d3_table *Converts Table to 'd3' Nodes and Links*

Description

Converts Table to 'd3' Nodes and Links

Usage

```
d3_table(tB = NULL, vars = NULL, agg = "Freq")
```

Arguments

tB	table to convert
vars	character vector of column names
agg	character column name of aggregated value

Value

list of two data.frames - one for nodes and one for links of the network. This structure is helpful when working with networkD3 and visNetwork.

Examples

```
library(d3r)
d3_table(Titanic, c("Class","Sex"))
```

d3_v8 *Create V8 Context with D3*

Description

Create V8 Context with D3

Usage

```
d3_v8(...)
```

Arguments

...	arguments passed to v8()

Value

v8 context with d3.js loaded and available as d3

Examples

```
## Not run:

# to do this all in R, please see ggraph
# https://github.com/thomasp85/ggraph
# by Thomas Lin Pedersen
library(d3r)

# make a simple data.frame of US state data
states <- data.frame(
  region = as.character(state.region),
  state = as.character(state.abb),
  population = state.x77[,"Population"],
  stringsAsFactors = FALSE
)

# use d3_nest to get the data.frame in a d3 hierarchy
state_hier <- d3_nest(
  states,
  value_cols = "population"
)

# use d3_v8 to do something useful with d3 and, our state data
ctx <- d3_v8()
ctx$eval(sprintf(
  " var states = %s",
  state_hier
))
ctx$eval(
"
// we assigned the variable states above
//  so now make it a real d3 hierarchy
var root = d3.hierarchy(states);

// sum on population
root.sum(function(d) {return d.population ? d.population : 0});


// use d3 to circle pack or state hierarchy
d3.pack()(root);

// get something we can convert into a data.frame in R
var states_packed = [];
root.each(function(d) {
  states_packed.push({
    name: d.data.name,
    radius: d.r,
    x: d.x,
    y: d.y
  });
});
"
```

```
)

# now get states_packed from our context
#  to plot in R
states_packed <- ctx$get("states_packed")
opar <- par(no.readonly=TRUE)
# make it square
par(pty="s")
symbols(
  states_packed$x,
  states_packed$y,
  states_packed$radius,
  inches=FALSE,
  asp=1
)
text(y~x, data=states_packed, labels=states_packed$name)
# return to original par before we made it square
par(opar)

# d3.quadtree example

library(d3r)

x = runif(100)
y = runif(100)

ctx <- d3_v8()
# assign pts as array of pts in V8
ctx$assign("pts", matrix(c(x,y),ncol=2,byrow=TRUE))
# use d3.quadtree() to plot rects
ctx$eval(
  "
  var d3q = d3.quadtree()
  .addAll(pts);
  // nodes function from https://bl.ocks.org/mbostock/4343214
  function nodes(quadtree) {
  var nodes = [];
  quadtree.visit(function(node, x0, y0, x1, y1) {
  nodes.push({x0:x0, y0:y0, x1: x1, y1: y1})
  });
  return nodes;
  }
  "
)

nodes <- ctx$get("nodes(d3q)", simplifyVector = FALSE)
# draw points
opar <- par(no.readonly=TRUE)
# make it square
par(pty="s")
plot(y~x)
# draw quadtree rects
rect(
```

```
  lapply(nodes,function(x){x$x0}),
  lapply(nodes,function(x){x$y0}),
  lapply(nodes,function(x){x$x1}),
  lapply(nodes,function(x){x$y1})
)
par(opar)

## End(Not run)
```

---

promote_na *Apply 'promote_na' to All Rows*

Description

Apply 'promote_na' to All Rows

Usage

```
promote_na(x)
```

Arguments

x	data.frame

Value

data.frame

---

promote_na_one *Promote NA to Top Level*

Description

Promote NA to Top Level

Usage

```
promote_na_one(x)
```

Arguments

x	data.frame

Value

data.frame

Index