Package 'networkD3'

July 22, 2025

Type Package

Title D3 JavaScript Network Graphs from R

Description Creates 'D3' 'JavaScript' network, tree, dendrogram, and Sankey graphs from 'R'.

Version 0.4.1

Date 2025-04-14

URL https://christophergandrud.github.io/networkD3/, https://github.com/christophergandrud/networkD3

BugReports https://github.com/christophergandrud/networkD3/issues

License GPL (>= 3)

Depends R (>= 3.0.0)

Imports data.tree, htmlwidgets (>= 0.3.2), igraph, jsonlite, magrittr

Suggests htmltools (>= 0.2.6), tibble

Encoding UTF-8

Enhances knitr, shiny

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author J.J. Allaire [aut],
Peter Ellis [ctb],
Christopher Gandrud [aut, cre],
Kevin Kuo [ctb],
B.W. Lewis [ctb],
Jonathan Owen [ctb],
Kenton Russell [aut],
Jennifer Rogers [ctb],
Charles Sese [ctb],
CJ Yetman [aut]

Maintainer Christopher Gandrud <christopher.gandrud@gmail.com>

Repository CRAN

Date/Publication 2025-04-14 16:20:02 UTC

Contents

`networkD3-package` *Tools for Creating D3 Network Graphs from R*

Description

Creates D3 JavaScript network, tree, dendrogram, and Sankey graphs from R.

as.radialNetwork *Convert an R hclust or dendrogram object into a radialNetwork list.*

Description

as.radialNetwork converts an R hclust or dendrogram object into a list suitable for use by the radialNetwork function.

Usage

```
as.radialNetwork(d, root)
```

Arguments

d	An object of R class hclust or dendrogram.
root	An optional name for the root node. If missing, use the first argument variable name.

Details

as.radialNetwork coverts R objects of class hclust or dendrogram into a list suitable for use with the radialNetwork function.

Examples

```
# Create a hierarchical cluster object and display with radialNetwork
## dontrun
hc <- hclust(dist(USArrests), "ave")
radialNetwork(as.radialNetwork(hc))
```

as_treenetdf *Convert one of numerous data types to treeNetwork's 'native' treenetdf form*

Description

The treeNetwork function uses a 'native' data format that consists of a data frame with minimally 2 vectors/columns, one named 'nodeId' and one named 'parentId'. Other columns in the data frame are also passed on to the JavaScript code and attached to the elements in the D3 visualization so that they can potentially be accessed by other JavaScript functions. This is an advantageous format because:

- it's an easy to use and understand R-like format
- a hierarchical network can be succinctly defined by a list of each unique node and its parent node

- since each row defines a unique node, additional columns can be added to add node-specific properties
- in a hierarchical network, every link/edge can be uniquely identified by the node which it leads to, therefore each link/edge can also be specifically addressed by adding columns for formatting of the incoming link

`as_treenetdf` can convert from any of the following data types:

- `leafpathdf` (table)–`parent|parent|node–data.frame`
- hierarchical nested list (JSON)
- `hclust`
- `data.tree` Node
- igraph
- ape `phylo`

Usage

```
as_treenetdf(data = NULL, ...)
```

Arguments

`data`	a tree network description in one of numerous forms (see details).
`...`	other arguments that will be passed on to as_treenetdf

Examples

```
links <- read.csv(header = TRUE, stringsAsFactors = FALSE, text = '
                  source,target,name
                  1,,one
                  2,1,two
                  3,1,three
                  4,1,four
                  5,2,five
                  6,2,six
                  7,2,seven
                  8,6,eight')

 # Convert data
 as_treenetdf(links, cols = c(nodeId = 'source', parentId = 'target'))

 # Graph (calls as_treenetdf internally)
 treeNetwork(links, cols = c(nodeId = 'source', parentId = 'target'))
```

as_treenetdf.data.frame *Convert a data.frame to a* treenetdf

Description

Convert a data.frame to a treenetdf

Usage

```
## S3 method for class 'data.frame'
as_treenetdf(
  data,
  cols = setNames(names(data), names(data)),
  df_type = "treenetdf",
  subset = names(data),
  root,
  ...
)
```

Arguments

data	a tree network description in one of numerous forms (see details).
cols	named character vector specifying the names of columns to be converted to the standard treenetdf names.
df_type	character specifying which type of data frame to convert. Can be treenetdf or leafpathdf.
subset	character vector specifying the names of the columns (in order) that should be used to define the hierarchy.
root	root name.
...	arguments to pass to methods.

as_treenetdf.hclust *Convert hclust objects to* treenetdf

Description

Convert hclust objects to treenetdf

Usage

```
## S3 method for class 'hclust'
as_treenetdf(data, ...)
```

Arguments

data	a tree network description in one of numerous forms (see details).
...	arguments to pass to methods.

---

as_treenetdf.igraph *Convert igraph tree to* treenetdf

Description

Convert igraph tree to treenetdf

Usage

```
## S3 method for class 'igraph'
as_treenetdf(data, root = "root", ...)
```

Arguments

data	a tree network description in one of numerous forms (see details).
root	character specifying the string that should be used to name the root node
...	arguments to pass to methods.

---

as_treenetdf.list *Convert a nested list to* treenetdf

Description

Convert a nested list to treenetdf

Usage

```
## S3 method for class 'list'
as_treenetdf(data, children_name = "children", node_name = "name", ...)
```

Arguments

data	a tree network description in one of numerous forms (see details).
children_name	character specifying the name used for the list element that contains the childeren elements.
node_name	character specifying the name used for the list element that contains the name of the node
...	arguments to pass to methods.

as_treenetdf.Node *data.tree to* treenetdf

Description

data.tree to treenetdf

Usage

```
## S3 method for class 'Node'
as_treenetdf(data, ...)
```

Arguments

data	a tree network description in one of numerous forms (see details).
...	arguments to pass to methods.

as_treenetdf.phylo *Phylo tree to* treenetdf

Description

Phylo tree to treenetdf

Usage

```
## S3 method for class 'phylo'
as_treenetdf(data, ...)
```

Arguments

data	a tree network description in one of numerous forms (see details).
...	arguments to pass to methods.

as_treenetdf.tbl_graph *tbl_graph_to_treenetdf*

Description

tbl_graph_to_treenetdf

Usage

```
## S3 method for class 'tbl_graph'
as_treenetdf(data, ...)
```

Arguments

data	a tree network description in one of numerous forms (see details).
...	arguments to pass to methods.

chordNetwork *Create Reingold-Tilford Tree network diagrams.*

Description

Create Reingold-Tilford Tree network diagrams.

Usage

```
chordNetwork(
  Data,
  height = 500,
  width = 500,
  initialOpacity = 0.8,
  useTicks = 0,
 colourScale = c("#1f77b4", "#aec7e8", "#ff7f0e", "#ffbb78", "#2ca02c", "#98df8a",
   "#d62728", "#ff9896", "#9467bd", "#c5b0d5", "#8c564b", "#c49c94", "#e377c2",
  "#f7b6d2", "#7f7f7f", "#c7c7c7", "#bcbd22", "#dbdb8d", "#17becf", "#9edae5"),
  padding = 0.1,
  fontSize = 14,
  fontFamily = "sans-serif",
  labels = c(),
  labelDistance = 30
)
```

Arguments

Data	A square matrix or data frame whose (n, m) entry represents the strength of the link from group n to group m
height	height for the network graph's frame area in pixels (if NULL then height is automatically determined based on context)
width	numeric width for the network graph's frame area in pixels (if NULL then width is automatically determined based on context)
initialOpacity	specify the opacity before the user mouses over the link
useTicks	integer number of ticks on the radial axis. The default is '0' which means no ticks will be drawn.
colourScale	specify the hexadecimal colours in which to display the different categories. If there are fewer colours than categories, the last colour is repeated as necessary (if NULL then defaults to D3 colour scale)
padding	specify the amount of space between adjacent categories on the outside of the graph
fontSize	numeric font size in pixels for the node text labels.
fontFamily	font family for the node text labels.
labels	vector containing labels of the categories
labelDistance	integer distance in pixels (px) between text labels and outer radius. The default is '30'.

Source

Mike Bostock: https://github.com/d3/d3/wiki/Chord-Layout.

Examples

```
## Not run:
#### Data about hair colour preferences, from
## https://github.com/mbostock/d3/wiki/Chord-Layout

hairColourData <- matrix(c(11975,  1951,  8010, 1013,
                           5871, 10048, 16145,  990,
                           8916,  2060,  8090,  940,
                           2868,  6171,  8045, 6907),
                           nrow = 4)

chordNetwork(Data = hairColourData,
             width = 500,
             height = 500,
             colourScale = c("#000000",
                             "#FFDD89",
                             "#957244",
                             "#F26223"),
             labels = c("red", "brown", "blond", "gray"))
```

```
## End(Not run)
```

`chordNetworkOutput` *Shiny bindings for networkD3 widgets*

Description

Output and render functions for using networkD3 widgets within Shiny applications and interactive Rmd documents.

Usage

```
chordNetworkOutput(outputId, width = "100%", height = "500px")

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

dendroNetworkOutput(outputId, width = "100%", height = "800px")

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

diagonalNetworkOutput(outputId, width = "100%", height = "800px")

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

forceNetworkOutput(outputId, width = "100%", height = "500px")

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

radialNetworkOutput(outputId, width = "100%", height = "800px")

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

sankeyNetworkOutput(outputId, width = "100%", height = "500px")

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

simpleNetworkOutput(outputId, width = "100%", height = "500px")

renderSimpleNetwork(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 networkD3 graph

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.

---

dendroNetwork *Create hierarchical cluster network diagrams.*

Description

Create hierarchical cluster network diagrams.

Usage

```
dendroNetwork(
  hc,
  height = 500,
  width = 800,
  fontSize = 10,
  linkColour = "#ccc",
  nodeColour = "#fff",
  nodeStroke = "steelblue",
  textColour = "#111",
  textOpacity = 0.9,
  textRotate = NULL,
  opacity = 0.9,
  margins = NULL,
  linkType = c("elbow", "diagonal"),
  treeOrientation = c("horizontal", "vertical"),
  zoom = FALSE
)
```

Arguments

hc	a hierarchical (hclust) cluster object.
height	height for the network graph's frame area in pixels
width	numeric width for the network graph's frame area in pixels
fontSize	numeric font size in pixels for the node text labels.
linkColour	character string specifying the colour you want the link lines to be. Multiple formats supported (e.g. hexadecimal).
nodeColour	character string specifying the colour you want the node circles to be. Multiple formats supported (e.g. hexadecimal).
nodeStroke	character string specifying the colour you want the node perimeter to be. Multiple formats supported (e.g. hexadecimal).

`textColour` character vector or scalar specifying the colour you want the text to be before they are clicked. Order should match the order of `hclust$labels`. Multiple formats supported (e.g. hexadecimal).

`textOpacity` numeric vector or scalar of the proportion opaque you would like the text to be before they are clicked. rder should match the order of `hclust$labels`.

`textRotate` numeric degress to rotate text for node text. Default is 0 for horizontal and 65 degrees for vertical.

`opacity` numeric value of the proportion opaque you would like the graph elements to be.

`margins` numeric value or named list of plot margins (top, right, bottom, left). Set the margin appropriately to accomodate long text labels.

`linkType` character specifying the link type between points. Options are 'elbow' and 'diagonal'.

`treeOrientation` character specifying the tree orientation, Options are 'vertical' and 'horizontal'.

`zoom` logical enabling plot zoom and pan

Source

Mike Bostock: https://bl.ocks.org/mbostock/4063570.

Fabio Nelli: https://www.meccanismocomplesso.org/en/dendrogramma-d3-parte1/

Examples

```
## Not run:
hc <- hclust(dist(USArrests), "ave")

dendroNetwork(hc, height = 600)
dendroNetwork(hc, treeOrientation = "vertical")

dendroNetwork(hc, height = 600, linkType = "diagonal")
dendroNetwork(hc, treeOrientation = "vertical", linkType = "diagonal")

dendroNetwork(hc, textColour = c("red", "green", "orange")[cutree(hc, 3)],
              height = 600)
dendroNetwork(hc, textColour = c("red", "green", "orange")[cutree(hc, 3)],
              treeOrientation = "vertical")

## End(Not run)
```

diagonalNetwork *Create Reingold-Tilford Tree network diagrams.*

Description

Create Reingold-Tilford Tree network diagrams.

Usage

```
diagonalNetwork(
  List,
  height = NULL,
  width = NULL,
  fontSize = 10,
  fontFamily = "serif",
  linkColour = "#ccc",
  nodeColour = "#fff",
  nodeStroke = "steelblue",
  textColour = "#111",
  opacity = 0.9,
  margin = NULL
)
```

Arguments

List	a hierarchical list object with a root node and children.
height	height for the network graph's frame area in pixels (if NULL then height is automatically determined based on context)
width	numeric width for the network graph's frame area in pixels (if NULL then width is automatically determined based on context)
fontSize	numeric font size in pixels for the node text labels.
fontFamily	font family for the node text labels.
linkColour	character string specifying the colour you want the link lines to be. Multiple formats supported (e.g. hexadecimal).
nodeColour	character string specifying the colour you want the node circles to be. Multiple formats supported (e.g. hexadecimal).
nodeStroke	character string specifying the colour you want the node perimeter to be. Multiple formats supported (e.g. hexadecimal).
textColour	character string specifying the colour you want the text to be before they are clicked. Multiple formats supported (e.g. hexadecimal).
opacity	numeric value of the proportion opaque you would like the graph elements to be.
margin	an integer or a named list/vector of integers for the plot margins. If using a named list/vector, the positions top, right, bottom, left are valid. If a single integer is provided, then the value will be assigned to the right margin. Set the margin appropriately to accomodate long text labels.

Source

Reingold. E. M., and Tilford, J. S. (1981). Tidier Drawings of Trees. IEEE Transactions on Software Engineering, SE-7(2), 223-228.

Mike Bostock: http://bl.ocks.org/mbostock/4339083.

Examples

```
## Not run:
#### Create tree from JSON formatted data
## Download JSON data
# Create URL. paste0 used purely to keep within line width.
URL <- paste0("https://cdn.rawgit.com/christophergandrud/networkD3/",
              "master/JSONdata//flare.json")

## Convert to list format
Flare <- jsonlite::fromJSON(URL, simplifyDataFrame = FALSE)

## Recreate Bostock example from http://bl.ocks.org/mbostock/4063550
diagonalNetwork(List = Flare, fontSize = 10, opacity = 0.9)

#### Create a tree dendrogram from an R hclust object
hc <- hclust(dist(USArrests), "ave")
diagonalNetwork(as.radialNetwork(hc))
diagonalNetwork(as.radialNetwork(hc), fontFamily = "cursive")

#### Create tree from a hierarchical R list
For an alternative structure see: http://stackoverflow.com/a/30747323/1705044
CanadaPC <- list(name = "Canada", children = list(list(name = "Newfoundland",
                     children = list(list(name = "St. John's"))),
                list(name = "PEI",
                     children = list(list(name = "Charlottetown"))),
                list(name = "Nova Scotia",
                     children = list(list(name = "Halifax"))),
                list(name = "New Brunswick",
                     children = list(list(name = "Fredericton"))),
                list(name = "Quebec",
                     children = list(list(name = "Montreal"),
                                     list(name = "Quebec City"))),
                list(name = "Ontario",
                     children = list(list(name = "Toronto"),
                                     list(name = "Ottawa"))),
                list(name = "Manitoba",
                     children = list(list(name = "Winnipeg"))),
                list(name = "Saskatchewan",
                     children = list(list(name = "Regina"))),
                list(name = "Nunavuet",
                     children = list(list(name = "Iqaluit"))),
                list(name = "NWT",
                     children = list(list(name = "Yellowknife"))),
                list(name = "Alberta",
                     children = list(list(name = "Edmonton"))),
                list(name = "British Columbia",
```

```
                    children = list(list(name = "Victoria"),
                                    list(name = "Vancouver"))),
               list(name = "Yukon",
                    children = list(list(name = "Whitehorse")))
))

diagonalNetwork(List = CanadaPC, fontSize = 10)

## End(Not run)
```

---

forceNetwork *Create a D3 JavaScript force directed network graph.*

Description

Create a D3 JavaScript force directed network graph.

Usage

```
forceNetwork(
  Links,
  Nodes,
  Source,
  Target,
  Value,
  NodeID,
  Nodesize,
  Group,
  height = NULL,
  width = NULL,
  colourScale = JS("d3.scaleOrdinal(d3.schemeCategory20);"),
  fontSize = 7,
  fontFamily = "serif",
  linkDistance = 50,
  linkWidth = JS("function(d) { return Math.sqrt(d.value); }"),
  radiusCalculation = JS(" Math.sqrt(d.nodesize)+6"),
  charge = -30,
  linkColour = "#666",
  opacity = 0.6,
  zoom = FALSE,
  legend = FALSE,
  arrows = FALSE,
  bounded = FALSE,
  opacityNoHover = 0,
  clickAction = NULL
)
```

Arguments

Links	a data frame object with the links between the nodes. It should include the Source and Target for each link. These should be numbered starting from 0. An optional Value variable can be included to specify how close the nodes are to one another.
Nodes	a data frame containing the node id and properties of the nodes. If no ID is specified then the nodes must be in the same order as the Source variable column in the Links data frame. Currently only a grouping variable is allowed.
Source	character string naming the network source variable in the Links data frame.
Target	character string naming the network target variable in the Links data frame.
Value	character string naming the variable in the Links data frame for how wide the links are.
NodeID	character string specifying the node IDs in the Nodes data frame.
Nodesize	character string specifying the a column in the Nodes data frame with some value to vary the node radius's with. See also radiusCalculation.
Group	character string specifying the group of each node in the Nodes data frame.
height	numeric height for the network graph's frame area in pixels.
width	numeric width for the network graph's frame area in pixels.
colourScale	character string specifying the categorical colour scale for the nodes. See https://github.com/d3/d3/blob/master/API.md#ordinal-scales.
fontSize	numeric font size in pixels for the node text labels.
fontFamily	font family for the node text labels.
linkDistance	numeric or character string. Either numberic fixed distance between the links in pixels (actually arbitrary relative to the diagram's size). Or a JavaScript function, possibly to weight by Value. For example: linkDistance = JS("function(d){return d.value * 10}").
linkWidth	numeric or character string. Can be a numeric fixed width in pixels (arbitrary relative to the diagram's size). Or a JavaScript function, possibly to weight by Value. The default is linkWidth = JS("function(d) { return Math.sqrt(d.value); }").
radiusCalculation	character string. A javascript mathematical expression, to weight the radius by Nodesize. The default value is radiusCalculation = JS("Math.sqrt(d.nodesize)+6").
charge	numeric value indicating either the strength of the node repulsion (negative value) or attraction (positive value).
linkColour	character vector specifying the colour(s) you want the link lines to be. Multiple formats supported (e.g. hexadecimal).
opacity	numeric value of the proportion opaque you would like the graph elements to be.
zoom	logical value to enable (TRUE) or disable (FALSE) zooming.
legend	logical value to enable node colour legends.
arrows	logical value to enable directional link arrows.

bounded	logical value to enable (TRUE) or disable (FALSE) the bounding box limiting the graph's extent. See http://bl.ocks.org/mbostock/1129492.
opacityNoHover	numeric value of the opacity proportion for node labels text when the mouse is not hovering over them.
clickAction	character string with a JavaScript expression to evaluate when a node is clicked.

Source

D3.js was created by Michael Bostock. See https://d3js.org/ and, more specifically for force directed networks https://github.com/d3/d3/blob/master/API.md#forces-d3-force.

See Also

JS.

Examples

```
# Load data
data(MisLinks)
data(MisNodes)
# Create graph
forceNetwork(Links = MisLinks, Nodes = MisNodes, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Group = "group", opacity = 0.4, zoom = TRUE)

# Create graph with legend and varying node radius
forceNetwork(Links = MisLinks, Nodes = MisNodes, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Nodesize = "size",
             radiusCalculation = "Math.sqrt(d.nodesize)+6",
             Group = "group", opacity = 0.4, legend = TRUE)

# Create graph directed arrows
forceNetwork(Links = MisLinks, Nodes = MisNodes, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Group = "group", opacity = 0.4, arrows = TRUE)

## Not run:
#### JSON Data Example
# Load data JSON formated data into two R data frames
# Create URL. paste0 used purely to keep within line width.
URL <- paste0("https://cdn.rawgit.com/christophergandrud/networkD3/",
              "master/JSONdata/miserables.json")

MisJson <- jsonlite::fromJSON(URL)

# Create graph
forceNetwork(Links = MisJson$links, Nodes = MisJson$nodes, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Group = "group", opacity = 0.4)
```

```
# Create graph with zooming
forceNetwork(Links = MisJson$links, Nodes = MisJson$nodes, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Group = "group", opacity = 0.4, zoom = TRUE)


# Create a bounded graph
forceNetwork(Links = MisJson$links, Nodes = MisJson$nodes, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Group = "group", opacity = 0.4, bounded = TRUE)

# Create graph with node text faintly visible when no hovering
forceNetwork(Links = MisJson$links, Nodes = MisJson$nodes, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Group = "group", opacity = 0.4, bounded = TRUE,
             opacityNoHover = TRUE)

## Specify colours for specific edges
# Find links to Valjean (11)
which(MisNodes == "Valjean", arr = TRUE)[1] - 1
ValjeanInds = which(MisLinks == 11, arr = TRUE)[, 1]

# Create a colour vector
ValjeanCols = ifelse(1:nrow(MisLinks) %in% ValjeanInds, "#bf3eff", "#666")

forceNetwork(Links = MisLinks, Nodes = MisNodes, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Group = "group", opacity = 0.8, linkColour = ValjeanCols)


## Create graph with alert pop-up when a node is clicked.  You're
# unlikely to want to do exactly this, but you might use
# Shiny.onInputChange() to allocate d.XXX to an element of input
# for use in a Shiny app.

MyClickScript <- 'alert("You clicked " + d.name + " which is in row " +
       (d.index + 1) +  " of your original R data frame");'

forceNetwork(Links = MisLinks, Nodes = MisNodes, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Group = "group", opacity = 1, zoom = FALSE,
             bounded = TRUE, clickAction = MyClickScript)

## End(Not run)
```

`igraph_to_networkD3` *Function to convert igraph graph to a list suitable for networkD3*

Description

Function to convert igraph graph to a list suitable for networkD3

Usage

```
igraph_to_networkD3(g, group, what = "both")
```

Arguments

g	an igraph class graph object
group	an object that contains node group values, for example, those created with igraph's membership function.
what	a character string specifying what to return. If what = 'links' or what = 'nodes' only the links or nodes are returned as data frames, respectively. If what = 'both' then both data frames will be return in a list.

Value

A list of link and node data frames or only the link or node data frames.

Examples

```
# Load igraph
library(igraph)

# Use igraph to make the graph and find membership
karate <- make_graph("Zachary")
wc <- cluster_walktrap(karate)
members <- membership(wc)

# Convert to object suitable for networkD3
karate_d3 <- igraph_to_networkD3(karate, group = members)

# Create force directed network plot
forceNetwork(Links = karate_d3$links, Nodes = karate_d3$nodes,
             Source = 'source', Target = 'target', NodeID = 'name',
             Group = 'group')

## Not run:
# Example with data from data frame
# Load data
## Original data from https://results.ref.ac.uk/DownloadSubmissions/ByUoa/21
data('SchoolsJournals')

# Convert to igraph
SchoolsJournals <- graph.data.frame(SchoolsJournals, directed = FALSE)

# Remove duplicate edges
SchoolsJournals <- simplify(SchoolsJournals)

# Find group membership
wt <- cluster_walktrap(SchoolsJournals, steps = 6)
members <- membership(wt)

# Convert igraph to list for networkD3
```

```
sj_list <- igraph_to_networkD3(SchoolsJournals, group = members)

# Plot as a forceDirected Network
forceNetwork(Links = sj_list$links, Nodes = sj_list$nodes, Source = 'source',
             Target = 'target', NodeID = 'name', Group = 'group',
             zoom = TRUE, linkDistance = 200)

## End(Not run)
```

JS *Create character strings that will be evaluated as JavaScript*

Description

Create character strings that will be evaluated as JavaScript

Usage

```
JS(...)
```

Arguments

...	character string to evaluate

Source

A direct import of JS from Ramnath Vaidyanathan, Yihui Xie, JJ Allaire, Joe Cheng and Kenton Russell (2015). htmlwidgets: HTML Widgets for R. R package version 0.4.

MisLinks *Les Miserables character links*

Description

A data file of links from Knuth's Les Miserables characters data base.

Usage

```
MisLinks
```

Format

A data set with 254 observations of 3 variables.

Source

See Mike Bostock https://bl.ocks.org/mbostock/4062045.

MisNodes *Les Miserables character nodes*

Description

A data file of nodes from Knuth's Les Miserables characters data base.

Usage

```
MisNodes
```

Format

A data set with 77 observations of 2 variables, plus made up node size variable.

Source

See Mike Bostock https://bl.ocks.org/mbostock/4062045.

pkg_installed *Check if a package is installed*

Description

Check if a package is installed

Usage

```
pkg_installed(pkg_name)
```

Arguments

pkg_name	character string name of package

`radialNetwork` *Create Reingold-Tilford Tree network diagrams.*

Description

Create Reingold-Tilford Tree network diagrams.

Usage

```
radialNetwork(
  List,
  height = NULL,
  width = NULL,
  fontSize = 10,
  fontFamily = "serif",
  linkColour = "#ccc",
  nodeColour = "#fff",
  nodeStroke = "steelblue",
  textColour = "#111",
  opacity = 0.9,
  margin = NULL
)
```

Arguments

`List`	a hierarchical list object with a root node and children.
`height`	height for the network graph's frame area in pixels (if `NULL` then height is automatically determined based on context)
`width`	numeric width for the network graph's frame area in pixels (if `NULL` then width is automatically determined based on context)
`fontSize`	numeric font size in pixels for the node text labels.
`fontFamily`	font family for the node text labels.
`linkColour`	character string specifying the colour you want the link lines to be. Multiple formats supported (e.g. hexadecimal).
`nodeColour`	character string specifying the colour you want the node circles to be. Multiple formats supported (e.g. hexadecimal).
`nodeStroke`	character string specifying the colour you want the node perimeter to be. Multiple formats supported (e.g. hexadecimal).
`textColour`	character string specifying the colour you want the text to be before they are clicked. Multiple formats supported (e.g. hexadecimal).
`opacity`	numeric value of the proportion opaque you would like the graph elements to be.
`margin`	an integer or a named `list`/`vector` of integers for the plot margins. If using a named `list`/`vector`, the positions `top`, `right`, `bottom`, `left` are valid. If a single integer is provided, then the value will be assigned to the right margin. Set the margin appropriately to accomodate long text labels.

Source

Reingold. E. M., and Tilford, J. S. (1981). Tidier Drawings of Trees. IEEE Transactions on Software Engineering, SE-7(2), 223-228.

Mike Bostock: http://bl.ocks.org/mbostock/4063550.

Examples

```
## Not run:
#### Create tree from JSON formatted data
## Download JSON data
# Create URL. paste0 used purely to keep within line width.
URL <- paste0("https://cdn.rawgit.com/christophergandrud/networkD3/",
              "master/JSONdata//flare.json")

## Convert to list format
Flare <- jsonlite::fromJSON(URL, simplifyDataFrame = FALSE)

## Recreate Bostock example from http://bl.ocks.org/mbostock/4063550
radialNetwork(List = Flare, fontSize = 10, opacity = 0.9)

#### Create a tree dendrogram from an R hclust object
hc <- hclust(dist(USArrests), "ave")
radialNetwork(as.radialNetwork(hc))
radialNetwork(as.radialNetwork(hc), fontFamily = "cursive")

#### Create tree from a hierarchical R list
For an alternative structure see: http://stackoverflow.com/a/30747323/1705044
CanadaPC <- list(name = "Canada", children = list(list(name = "Newfoundland",
                     children = list(list(name = "St. John's"))),
                list(name = "PEI",
                     children = list(list(name = "Charlottetown"))),
                list(name = "Nova Scotia",
                     children = list(list(name = "Halifax"))),
                list(name = "New Brunswick",
                     children = list(list(name = "Fredericton"))),
                list(name = "Quebec",
                     children = list(list(name = "Montreal"),
                                     list(name = "Quebec City"))),
                list(name = "Ontario",
                     children = list(list(name = "Toronto"),
                                     list(name = "Ottawa"))),
                list(name = "Manitoba",
                     children = list(list(name = "Winnipeg"))),
                list(name = "Saskatchewan",
                     children = list(list(name = "Regina"))),
                list(name = "Nunavuet",
                     children = list(list(name = "Iqaluit"))),
                list(name = "NWT",
                     children = list(list(name = "Yellowknife"))),
                list(name = "Alberta",
                     children = list(list(name = "Edmonton"))),
                list(name = "British Columbia",
```

```
                    children = list(list(name = "Victoria"),
                                    list(name = "Vancouver"))),
               list(name = "Yukon",
                    children = list(list(name = "Whitehorse")))
))

radialNetwork(List = CanadaPC, fontSize = 10)

## End(Not run)
```

---

sankeyNetwork *Create a D3 JavaScript Sankey diagram*

Description

Create a D3 JavaScript Sankey diagram

Usage

```
sankeyNetwork(
  Links,
  Nodes,
  Source,
  Target,
  Value,
  NodeID,
  NodeGroup = NodeID,
  LinkGroup = NULL,
  units = "",
  colourScale = JS("d3.scaleOrdinal(d3.schemeCategory20);"),
  fontSize = 7,
  fontFamily = NULL,
  nodeWidth = 15,
  nodePadding = 10,
  margin = NULL,
  height = NULL,
  width = NULL,
  iterations = 32,
  sinksRight = TRUE
)
```

Arguments

Links	a data frame object with the links between the nodes. It should have include the Source and Target for each link. An optional Value variable can be included to specify how close the nodes are to one another.

Nodes	a data frame containing the node id and properties of the nodes. If no ID is specified then the nodes must be in the same order as the Source variable column in the Links data frame. Currently only grouping variable is allowed.
Source	character string naming the network source variable in the Links data frame.
Target	character string naming the network target variable in the Links data frame.
Value	character string naming the variable in the Links data frame for how far away the nodes are from one another.
NodeID	character string specifying the node IDs in the Nodes. data frame. Must be 0-indexed.
NodeGroup	character string specifying the node groups in the Nodes. Used to color the nodes in the network.
LinkGroup	character string specifying the groups in the Links. Used to color the links in the network.
units	character string describing physical units (if any) for Value
colourScale	character string specifying the categorical colour scale for the nodes. See https://github.com/d3/d3/blob/master/API.md#ordinal-scales.
fontSize	numeric font size in pixels for the node text labels.
fontFamily	font family for the node text labels.
nodeWidth	numeric width of each node.
nodePadding	numeric essentially influences the width height.
margin	an integer or a named list/vector of integers for the plot margins. If using a named list/vector, the positions top, right, bottom, left are valid. If a single integer is provided, then the value will be assigned to the right margin. Set the margin appropriately to accomodate long text labels.
height	numeric height for the network graph's frame area in pixels.
width	numeric width for the network graph's frame area in pixels.
iterations	numeric. Number of iterations in the diagramm layout for computation of the depth (y-position) of each node. Note: this runs in the browser on the client so don't push it too high.
sinksRight	boolean. If TRUE, the last nodes are moved to the right border of the plot.

Source

D3.js was created by Michael Bostock. See https://d3js.org/ and, more specifically for Sankey diagrams https://bost.ocks.org/mike/sankey/.

See Also

JS

Examples

```
## Not run:
# Recreate Bostock Sankey diagram: https://bost.ocks.org/mike/sankey/
# Load energy projection data
URL <- paste0('https://cdn.rawgit.com/christophergandrud/networkD3/',
              'master/JSONdata/energy.json')
energy <- jsonlite::fromJSON(URL)

# Plot
sankeyNetwork(Links = energy$links, Nodes = energy$nodes, Source = 'source',
             Target = 'target', Value = 'value', NodeID = 'name',
             units = 'TWh', fontSize = 12, nodeWidth = 30)

# Colour links
energy$links$energy_type <- sub(' .*', '',
                                energy$nodes[energy$links$source + 1, 'name'])

sankeyNetwork(Links = energy$links, Nodes = energy$nodes, Source = 'source',
             Target = 'target', Value = 'value', NodeID = 'name',
             LinkGroup = 'energy_type', NodeGroup = NULL)


## End(Not run)
```

`saveNetwork` *Save a network graph to an HTML file*

Description

Save a networkD3 graph to an HTML file for sharing with others. The HTML can include it's dependencies in an adjacent directory or can bundle all dependencies into the HTML file (via base64 encoding).

Usage

```
saveNetwork(network, file, selfcontained = TRUE)
```

Arguments

`network`	Network to save (e.g. result of calling the function `simpleNetwork`).
`file`	File to save HTML into
`selfcontained`	Whether to save the HTML as a single self-contained file (with external resources base64 encoded) or a file with external resources placed in an adjacent directory.

SchoolsJournals — *Edge list of REF (2014) journal submissions for Politics and International Relations*

Description

Edge list of REF (2014) journal submissions for Politics and International Relations

Usage

```
SchoolsJournals
```

Format

A data set with 2732 rows and 3 variables.

Source

See REF 2014 https://results.ref.ac.uk/DownloadSubmissions/ByUoa/21.

simpleNetwork — *Function for creating simple D3 JavaScript force directed network graphs.*

Description

`simpleNetwork` creates simple D3 JavaScript force directed network graphs.

Usage

```
simpleNetwork(
  Data,
  Source = 1,
  Target = 2,
  height = NULL,
  width = NULL,
  linkDistance = 50,
  charge = -30,
  fontSize = 7,
  fontFamily = "serif",
  linkColour = "#666",
  nodeColour = "#3182bd",
  opacity = 0.6,
  zoom = F
)
```

Arguments

Data	a data frame object with three columns. The first two are the names of the linked units. The third records an edge value. (Currently the third column doesn't affect the graph.)
Source	character string naming the network source variable in the data frame. If Source = NULL then the first column of the data frame is treated as the source.
Target	character string naming the network target variable in the data frame. If Target = NULL then the second column of the data frame is treated as the target.
height	height for the network graph's frame area in pixels (if NULL then height is automatically determined based on context)
width	numeric width for the network graph's frame area in pixels (if NULL then width is automatically determined based on context)
linkDistance	numeric distance between the links in pixels (actually arbitrary relative to the diagram's size).
charge	numeric value indicating either the strength of the node repulsion (negative value) or attraction (positive value).
fontSize	numeric font size in pixels for the node text labels.
fontFamily	font family for the node text labels.
linkColour	character string specifying the colour you want the link lines to be. Multiple formats supported (e.g. hexadecimal).
nodeColour	character string specifying the colour you want the node circles to be. Multiple formats supported (e.g. hexadecimal).
opacity	numeric value of the proportion opaque you would like the graph elements to be.
zoom	logical value to enable (TRUE) or disable (FALSE) zooming

Source

D3.js was created by Michael Bostock. See https://d3js.org/ and, more specifically for directed networks https://github.com/d3/d3/blob/master/API.md#forces-d3-force

Examples

```
# Fake data
Source <- c("A", "A", "A", "A", "B", "B", "C", "C", "D")
Target <- c("B", "C", "D", "J", "E", "F", "G", "H", "I")
NetworkData <- data.frame(Source, Target)

# Create graph
simpleNetwork(NetworkData)
simpleNetwork(NetworkData, fontFamily = "sans-serif")
```

`treeNetwork` *Create collapsible tree network diagrams.*

Description

Create collapsible tree network diagrams.

Usage

```
treeNetwork(
  data,
  width = NULL,
  height = NULL,
  treeType = "tidy",
  direction = "right",
  linkType = "diagonal",
  defaults = NULL,
  mouseover = "",
  mouseout = "",
  inbrowser = FALSE,
  ...
)
```

Arguments

`data`	a tree network description in one of numerous forms (see details)
`width`	numeric width for the network graph's frame area in pixels
`height`	height for the network graph's frame area in pixels
`treeType`	character specifying the tree layout type. Options are 'tidy' and 'cluster'.
`direction`	character specifying the direction in which the tree layout shoud grow. One of 'right', 'left', 'down', 'up', or 'radial'
`linkType`	character specifying the link type between points. Options are 'elbow' and 'diagonal'.
`defaults`	named character vector specifying custom default node and link formatting options
`mouseover`	character specifying JavaScript code to be run on mouseover events
`mouseout`	character specifying JavaScript code to be run on mouseout events
`inbrowser`	logical specifying to open the plot in a new browser window
`...`	other arguments that will be passed on to as_treenetdf

`treeNetwork-shiny` *Shiny bindings for treeNetwork*

Description

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

Usage

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

renderTreeNetwork(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 treeNetwork
`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.

Index