Package ‘networkGraphsR’

December 3, 2024

Title A set of functions that provide tools for creating interactive network visualisations

Version 0.0.2

Description A set of functions that provide tools for creating interactive network visualisations and managing categorical group data. It simplifies the process of analysing relationships and connections between entities such as initiatives, individuals, institutions, and associated attributes. The package offers customisable visualisations with adjustable node sizes, link distances, colours, and interactivity, making it ideal for exploratory data analysis and presentation of complex relational datasets.

License LGPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Depends R (>= 2.10)

Imports magrittr

Contents

`individuals_by` *"individuals_by"*

Description

individuals_by function creates a force-directed network visualization using the forceNetwork function from the networkD3 package.

Usage

```
individuals_by(
  individual_data = individuals,
  ind_id = nom,
  group = NULL,
  group_other = NULL,
  group_other_name = NULL,
  font_size = 7,
  height = NULL,
  width = NULL,
  colour_scale = JS("d3.scaleOrdinal(d3.schemeCategory20);"),
  font_family = "serif",
  link_distance = 50,
  link_width = JS("function(d) { return Math.sqrt(d.value); }"),
  radius_calculation = "4*Math.sqrt(d.nodesize)+2",
  charge = -30,
  link_colour = "#666",
  opacity = 0.6,
  zoom = FALSE,
  arrows = FALSE,
  bounded = FALSE,
  display_labels = 0,
  click_action = NULL
)
```

Arguments

`individual_data`
The data frame containing the individual-level data. By default, it assumes a data frame called `individuals`.

`ind_id`
The variable/column in the `individual_data` data frame that represents the individual identifier. By default, it assumes a column named "nom".

`group`
The variable/column in the `individual_data` data frame that represents the grouping variable. It is used to assign colors to the nodes in the visualization. If not provided, the visualization will not group the nodes.

`group_other`
is a string representing the name of the column in the "individual_data" data block that contains additional group information for each individual.

`group_other_name`
is a string representing a custom name for the "group_other" column to display in the plot.

`font_size`
numeric font size in pixels for the node text labels.By default, it is set to 7.

`height`
numeric height for the network graph's frame area in pixels.

`width`
numeric width for the network graph's frame area in pixels.

`colour_scale`
character string specifying the categorical colour scale for the nodes. See `https://github.com/d3/c`

`font_family`
font family for the node text labels.

`link_distance`
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").

`link_width` 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); ").

`radius_calculation` 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).

`link_colour` 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.

`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`.

`display_labels` is a numeric value representing the number of characters of the label to display on each node.

`click_action` character string with a JavaScript expression to evaluate when a node is clicked.

Value

Returns a network graph object

Examples

```
# TODO
```

`initiative_by` *"Initiative_by"*

Description

This function is a wrapper function for `networkD3::forceNetwork`. explain what it does -

Usage

```
initiative_by(
  initiative_data,
  by = "pays",
  sep = ".",
  filter_var = NULL,
  filter_vals = NULL,
  node_size = c("type", "age"),
  group = NULL,
  font_size = 7,
  height = NULL,
  width = NULL,
  colour_scale = JS("d3.scaleOrdinal(d3.schemeCategory20);"),
  font_family = "serif",
  link_distance = 50,
  link_width = JS("function(d) { return Math.sqrt(d.value); }"),
  radius_calculation = "4*Math.sqrt(d.nodesize)+2",
  charge = -30,
  link_colour = "#666",
  opacity = 0.6,
  zoom = FALSE,
  arrows = FALSE,
  bounded = FALSE,
  display_labels = 0,
  click_action = NULL
)
```

Arguments

`initiative_data`
a data frame object containing the initiative data

`by` the values around which nodes are formed

`filter_var` filtered variable

`filter_vals` values associated to the filter

`node_size` character string specifying the a column in the `initiative_data` 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 `initiative_data` data frame.

`font_size` numeric font size in pixels for the node 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.

`colour_scale` character string specifying the categorical colour scale for the nodes. See https://github.com/d3/d3/blo scales.

`font_family` font family for the node text labels.

`link_distance` 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").

`link_width` 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); ").

`radius_calculation`
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).

`link_colour` 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.

`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.
`display_labels`	
`click_action`	character string with a JavaScript expression to evaluate when a node is clicked.

Value

Returns a network graph object

Examples

```
# todo
```

`map_data` *Map Data for Network Visualisation*

Description

This function takes a data frame and generates a network visualization using the `forceNetwork` function from the `networkD3` package. It separates and expands the data based on the specified values and node values, and creates links between them.

Usage

```
map_data(data, values, node_values, colour_values = NULL)
```

Arguments

`data`	The original data frame.
`values`	The column name representing the primary values to be mapped.
`node_values`	A character vector specifying the column names representing the node values.

Value

A network visualisation generated using the `forceNetwork` function.

Examples

```
# An example with simple data
df_original <- data.frame(initiative = c("A", "B", "C", "D"),
                          country = c("BF", "BF Niger", "Niger", "Niger"),
                          ex = c("a b", "a b", "a", "b"))
map_data(df_original, "initiative", "country")
map_data(df_original, "initiative", "country", "ex")
map_data(df_original, "initiative", c("country", "ex"), "ex")


df_original2 <- data.frame(individuals = c("A", "B", "C", "D"),
                           institutions = c("a b", "a", "a", "b c"),
                           initiatives = c("z", "z", "y z", "y"))
map_data(df_original2, "individuals", c("institutions", "initiatives"))
map_data(df_original2, "individuals", c("institutions", "initiatives"), "institutions")
```

replace_other *"replace_other"*

Description

This function replaces a specific group in a data frame with another group.

Usage

```
replace_other(
  data = individuals,
  group = NULL,
  group_other = paste0("autre_", group),
  group_other_name = NULL,
  RAS = FALSE
)
```

Arguments

`data` The data frame in which the replacement will be performed. By default, it assumes a data frame called `individuals`.

`group` The name of the group variable to be replaced. This parameter is required.

`group_other` The name of the replacement group variable. It is constructed by appending the prefix "autre_" to the original group name. For example, if `group` is "pays", then `group_other` becomes "autre_pays".

`group_other_name` The specific value within the `group` variable that should be replaced. When this value is encountered in the `group` variable, it will be replaced with the corresponding value from `group_other`. This parameter is required.

`RAS` A logical value indicating whether the replacement should be performed with "RAS" when the `group_other_name` is encountered. If `RAS` is set to `TRUE`, the replacement will be "RAS"; otherwise, it will be the corresponding value from `group_other`. By default, it is set to `FALSE`.

Value

Returns data

Note

The `mutate` function is used to create a new column in the data frame named `group_var`, which replaces the values based on the provided conditions. When `RAS` is set to `TRUE`, it checks if the value in the `group` variable is equal to `group_other_name`. If it is, the corresponding value from `group_other` is assigned to `group_var`. Additionally, if the value in the `group` variable is `NA` (missing), it is replaced with "RAS". If `RAS` is set to `FALSE`, the replacement is performed without using "RAS".

Examples

```
# TODO
```

Index