Package ‘sgraph’
August 19, 2024

Title  Network Visualization Using 'sigma.js'
Version 1.1.0

Description  Interactive visualizations of graphs created with the 'igraph' package using a 'htmlwidgets' wrapper for the 'sigma.js' network visualization v2.4.0 <https://www.sigmajs.org/>, enabling to display several thousands of nodes. While several 'R' packages have been developed to interface 'sigma.js', all were developed for v1.x.x and none have migrated to v2.4.0 nor are they planning to. This package builds upon the 'sigmaNet' package, and users familiar with it will recognize the similar design approach. Two extensions have been added to the classic 'sigma.js' visualizations by overriding the underlying 'JavaScript' code, enabling to draw a frame around node labels, and to display labels on multiple lines by parsing line breaks. Other additional functionalities that did not require overriding 'sigma.js' code include toggling node visibility when clicked using a node attribute and highlighting specific edges. 'sigma.js' is currently preparing a stable release v3.0.0, and this package plans to update to it when it is available.

Imports  cowplot, ggplot2, grDevices, htmlwidgets, igraph, jsonlite, magrittr, RColorBrewer, stringi

Depends  R (>= 3.5.0)

Suggests  knitr, testthat

VignetteBuilder  knitr

License  GPL-3

Encoding  UTF-8

RoxygenNote  7.3.1

URL  https://gitlab.com/thomaschln/sgraph

BugReports  https://gitlab.com/thomaschln/sgraph/-/issues

LazyData  true

NeedsCompilation  no


Maintainer  Thomas Charlon <charlon@protonmail.com>

Repository  CRAN

Date/Publication  2024-08-19 16:00:02 UTC
**Contents**

- add_edge_color ................................................. 2
- add_edge_size .................................................. 3
- add_edge_zindex ............................................... 4
- add_igraph_info ............................................... 5
- add_listener ................................................... 5
- add_node_hidden ............................................... 6
- add_node_labels ............................................... 6
- add_node_size ................................................ 7
- convert_to_spring_weights ................................. 8
- get_color_map ................................................. 9
- get_legend ..................................................... 9
- highlight_multiple_connected ............................. 10
- kgraph_to_igraph .............................................. 10
- lesMis .......................................................... 11
- l_graph_to_igraph ............................................ 11
- multiline_labels ............................................. 12
- renderSgraph .................................................. 13
- scale_graph ................................................... 13
- sgraphOutput .................................................. 14
- sgraph_clusters ............................................... 15
- sigma_from_igraph ........................................... 16
- %<>% ............................................................ 17
- %$% ............................................................. 17
- %>>% ............................................................. 18

**Index**

---

**add_edge_color**

*Modify the edge colors of a sgraph object.*

---

**Description**

Modify the edge colors of a sgraph object by providing a single color. Also works with a vector of correct size.

**Usage**

```r
add_edge_color(
  sigma_obj,
  one_color = NULL,
  color_attr = NULL,
  color_palette = "Set2"
)
```
add_edge_size

Arguments

sigma_obj  sgraph object
one_color   A single color to color all of the nodes (hex format)
color_attr  The name of an edge attribute
color_palette Name of RColorBrewer palette to use

Value

sgraph with modified edge colors

Examples

library(igraph)
library(sgraph)
data(lesMis)
sig <- sigma_from_igraph(igraph = lesMis) %>%
    add_edge_color(one_color = "#ccc")
sig

add_edge_size

Modify the edge size of a sgraph object.

Description

Modify the edge size of a sgraph by providing a single size

Usage

add_edge_size(sigma_obj, one_size = NULL)

Arguments

sigma_obj  sgraph object
one_size   A single size to use for all edges

Value

sgraph with modified edge sizes
Examples

```r
library(igraph)
library(sgraph)

data(lesMis)

sig <- sigma_from_igraph(igraph = lesMis) %>%
  add_edge_size(one_size = 5)

add_edge_zindex(sig, zindex = 2)
```

Description

Modify the edge zIndex of a sgraph object.

Usage

```r
add_edge_zindex(sigma_obj, zindex)
```

Arguments

- **sigma_obj**: sgraph object
- **zindex**: Zindex value, larger is drawn above.

Value

sgraph

Examples

```r
library(igraph)
library(sgraph)

data(lesMis)

sig <- sigma_from_igraph(igraph = lesMis) %>%
  add_edge_zindex(zindex = 2)

sig
```
add_igraph_info

Add nodes information to the igraph object

Description
Modify the node attributes of an existing igraph object by providing a dataframe

Usage
add_igraph_info(igraph, df_nodes, fields = names(df_nodes))

Arguments
igraph Igraph object to modify
df_nodes Data frame to add to nodes
fields Columns of df_nodes to add. First must be the node identifier.

Value
A sgraph object with modified node labels

Examples
library(igraph)
library(sgraph)
data(lesMis)

df_nodes = cbind.data.frame(name = igraph::vertex_attr(lesMis, 'label'),
log10_degree = degree(lesMis))

igraph = add_igraph_info(lesMis, df_nodes)

sig <- sigma_from_igraph(lesMis) %>%
  add_node_size(size_vector = 'log10_degree')

add_listener

Add a listener

Description
Add a listener to report data from a 'sgraph' object in 'Shiny' back to the R session.

Usage
add_listener(sigma_obj, listener)
**add_node_labels**

**Arguments**

- **sigma_obj**: Sgraph object, created using the `sigma_from_igraph` function
- **listener**: Either "clickNode" to listen to node clicks or "hoverNode" to listen to node hover

**Value**

Sgraph object with listener

**Description**

Modify the node labels of an existing sgraph object by providing an attribute from the initial igraph to use as the labels.

**Usage**

```r
add_node_labels(sigma_obj, label_attr = NULL)
```

**Arguments**

- **sigma_obj**: sgraph object, returned by `sigma_from_igraph` function
- **label_attr**: Attribute to use to replace node labels

**Value**

Sgraph object with modified node hidden attribute

**Description**

Modify the node visibility of a sgraph object.

**Usage**

```r
add_node_hidden(sigma_obj, hidden_attr)
```

**Arguments**

- **sigma_obj**: sgraph object, returned by `sigma_from_igraph` function
- **hidden_attr**: Attribute to use to set node hidden value
add_node_size

Value

sgraph object with modified node labels

Examples

```r
library(igraph)
library(sgraph)
data(lesMis)
sig <- sigma_from_igraph(igraph = lesMis) %>%
  add_node_labels(label_attr = 'label')
sig
```

---

**add_node_size**       Modify the node size of a sgraph object.

Description

Modify the node size of an existing sgraph object by providing either: (1) A single size to use for all nodes; (2) a vector of node sizes; or (3) a metric to use to scale the nodes.

Usage

```r
add_node_size(
  sigma_obj,  # sgraph object, returned by sigma_from_igraph function
  min_size = 1,  # Minimum node size on the graph (for scaling)
  max_size = 3,  # Maximum node size on the graph (for scaling)
  one_size = NULL,  # A single size to use for all nodes
  size_vector = NULL  # An optional vector with the sizes for each node
)
```

Arguments

- `sigma_obj`: sgraph object, returned by `sigma_from_igraph` function
- `min_size`: Minimum node size on the graph (for scaling)
- `max_size`: Maximum node size on the graph (for scaling)
- `one_size`: A single size to use for all nodes
- `size_vector`: An optional vector with the sizes for each node

Value

A sgraph object with modified node sizes
Examples

```r
library(igraph)
library(sgraph)

data(lesMis)

layout <- layout_nicely(lesMis)

# one size for all nodes
sig <- sigma_from_igraph(igraph = lesMis, layout = layout) %>%
  add_node_size(one_size = 3)
sig

# using a vector
custom_size <- log10(degree(lesMis))
sig <- sigma_from_igraph(igraph = lesMis, layout = layout) %>%
  add_node_size(size_vector = custom_size)
sig
```

---

**convert_to_spring_weights**

*Convert weights for spring layout*

### Description

Apply spring weights (revert weights). Can add weak links to selected nodes, but best without.

### Usage

```r
convert_to_spring_weights(df_links, selected_nodes = NULL)
```

### Arguments

- `df_links`  
  Links data frame of a sgraph object
- `selected_nodes`  
  Nodes identifiers to be used for the selection.

### Value

Links data frame of a sgraph object
get_color_map

Build a color map

Description
Build a color map

Usage
get_color_map(colors, palette = RColorBrewer::brewer.pal(8, "Dark2"))

Arguments
- colors: Groups that will be assigned to colors
- palette: Palette to use, typically a RColorBrewer palette

Value
Data frame mapping group names to colors

get_legend

Get the legend for a sgraph network

Description
Get the legend for a sgraph network

Usage
get_legend(colors_map, clusters)

Arguments
- colors_map: Color mapping to use, typically built by 'get_color_map'
- clusters: Group names

Value
Ggplot object
highlight_multiple_connected

*Highlight edges of multiple connected nodes*

**Description**

Using a selection of nodes, highlight edges linking to nodes that are connected to several nodes from the selection. Differentiate multiple connected and fully connected (all selected nodes). Use a maximum number of connected nodes to use lighter colored edges (default 20).

**Usage**

```r
df_links, selected_nodes, n_max = 20,
  dark_cols = c("#ddd", ",#444", ",#444"),
  light_cols = c("#efefef", ",#ddd", ",#bbb")
)
```

**Arguments**

- `df_links` Links data frame of a sgraph object
- `selected_nodes` Nodes identifiers to be used for the selection.
- `n_max` Maximum number of connected nodes, to use either lighter or darker color sets for edges (default 20).
- `dark_cols` Three hex values for colors to use with `n_max`.
- `light_cols` Three hex values for colors to use with `n_max`.

**Value**

Links data frame of a sgraph object

---

**kgraph_to_lgraph**

*Kgraph fit to graph list object*

**Description**

Example function to build a graph list object (list of nodes and links data frames) from a kgraph object (embeddings with cosine similarity cut-off based on random null concept pairs and known related concept pairs)

**Usage**

```r
df_links, selected_nodes

kgraph_to_lgraph(l_fit_embeds)
```
### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>l_fit_embeds</code></td>
<td>kgraph object: embeddings with cosine similarity cut-off based on random null concept pairs and known related concept pairs</td>
</tr>
</tbody>
</table>

### Value

Graph list object: list of nodes and links data frames

---

**lesMis**  
*Co-appearances of characters in "Les Miserables"*

### Description

A graph where the nodes are characters in "Les Miserables" and the edges are times that the characters appeared together in the novel.

### Usage

```
lesMis
```

### Format

An igraph object with 77 nodes and 254 edges

- **id** numeric id of nodes
- **label** character label (names) of nodes
- **value** numeric weight of the edges (number of co-appearances)

### Source


---

**l_graph_to_igraph**  
*Graph list object to igraph object*

### Description

Build an igraph object from a graph list object (list of nodes and links data frames)

### Usage

```
l_graph_to_igraph(l_graph)
```
multiline_labels

**Arguments**

- `l_graph`: graph list object: list of nodes and links data frames

**Value**

igraph object

---

**multiline_labels**

*Format multiline labels*

**Description**

The sgraph R package extends the sigma.js library to enable multiline labels. The Javascript functions will start new lines on line breaks (`\n`) and this function enables to easily format the details of nodes by showing the value of the relationship (edge weights) it has with other nodes. See the Shiny examples for use cases.

**Usage**

```r
multiline_labels(
  df_nodes,
  display_val_str = "\nP-value: ",
  replace_codes = TRUE,
  label_str = "Label: ",
  group_str = "Group: 
)
```

**Arguments**

- `df_nodes`: Nodes data frame of a sgraph object
- `display_val_str`: String that will be prepended to each edge weight
- `replace_codes`: Should the label of the node replace the id?
- `label_str`: String that will be prepended to the node label
- `group_str`: String that will be prepended to the node group

**Value**

sgraph object with modified node hidden attribute
renderSgraph: Render a sgraph visualization in Shiny

Description

Render a sgraph visualization in Shiny

Usage

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

Arguments

expr
An expression that creates a sgraph visualization

env
Defaults to parent.frame() (cf. Shiny docs)

quoted
Defaults to FALSE (cf. Shiny docs)

Value

Htmlwidgets render object

cscale_graph: Scale weights

Description

First try to linearize the weights with the best logarithmic and polynomial, then perform exponential scaling and set upper and lower bounds.

Usage

scale_graph(
  weights,
  exp_scale = exp(1),
  upper_bound_mult = 25,
  lower_bound_const = 5
)
Arguments
weights Either nodes or links weights vector
exp_scale Scale for exponential transform
upper_bound_mult Constant to multiply weights by after scaling. Use to set an upper bound for weights.
lower_bound_const Constant to set a lower bound for weights. All weights below will be set to lower bound.

Value
Weights vector

---

`sgraphOutput`  
Create a UI element for a sgraph visualization in Shiny

Description
Create a UI element for a sgraph visualization in Shiny

Usage
`sgraphOutput(outputId, width = "100\%", height = "400px")`

Arguments
outputId ID of the UI element
width Width of the UI element
height Height of the UI element

Value
Htmlwidgets output object
sgraph_clusters

**Build a sgraph object colored by clusters**

**Description**

Wrapper function to build a sigma.js visualization of an igraph object and color it using a `clusters` attribute. It calls the sigma_from_igraph function, and manages labels, node sizes, color mapping, layouts, and optionally arrows. The clusters attribute can be also be disabled to just use the wrapper to manage the other attributes.

**Usage**

```r
sgraph_clusters(
  igraph,
  color_map = NULL,
  label = "name",
  clusters = TRUE,
  arrows = FALSE,
  node_size = NULL,
  layout = igraph::layout_with_fr(igraph),
  ...
)
```

**Arguments**

- **igraph** Igraph object
- **color_map** Use a color mapping to select colors (enables to link the graph with other plots) as returned by the get_color_map function. Leave NULL for automatic colors (default).
- **label** Name of the igraph attribute to use as labels
- **clusters** Whether or not to use a column named clusters to color the nodes.
- **arrows** Whether or not to display arrows on directed edges.
- **node_size** Passed to add_node_size function, either a numeric or an attribute name. Default is NULL.
- **layout** Output of an igraph layout (default: layout_with_fr)
- **...** Passed to sigma_from_igraph

**Value**

Htmlwidget object
Description

Basic sigma.js visualization of an igraph object, with pipeable syntax.

Usage

sigma_from_igraph(igraph, layout = NULL, label_color = "#fff", width = "100%", height = "400px", elementId = NULL, label_grid_cell_size = 200)

Arguments

igraph Igraph object
layout Output of an igraph layout (default: layout_nicely)
label_color Hex color for labels
width Width of the output graph (default: fit container)
height Height of the output graph (default: fit container)
elemId Do not specify, used by the htmlwidgets package
label_grid_cell_size Sigma.js corresponding parameter. Roughly goes from 1 to 5000, the smaller the more labels displayed.

Value

Htmlwidget object, meant to be called directly to render a default visualization, or passed to other functions to change attributes (colors, sizes, interactivity, etc.).

Examples

library(sgraph)
data(lesMis)

sig <- sigma_from_igraph(igraph = lesMis)
sig
Assignment pipe

Description
Pipe an object forward into a function or call expression and update the ‘lhs’ object with the resulting value. Magrittr imported function, see details and examples in the magrittr package.

Arguments
- **lhs**: An object which serves both as the initial value and as target.
- **rhs**: a function call using the magrittr semantics.

Value
None, used to update the value of lhs.

Exposition pipe

Description
Expose the names in ‘lhs’ to the ‘rhs’ expression. Magrittr imported function, see details and examples in the magrittr package.

Arguments
- **lhs**: A list, environment, or a data.frame.
- **rhs**: An expression where the names in lhs is available.

Value
Result of rhs applied to one or several names of lhs.
Pipe

Description
Pipe an object forward into a function or call expression. Magrittr imported function, see details and examples in the magrittr package.

Arguments
- `lhs` A value or the magrittr placeholder.
- `rhs` A function call using the magrittr semantics.

Value
Result of `rhs` applied to `lhs`, see details in magrittr package.
Index

* datasets
  lesMis, 11
  %<>%, 17
  %>%, 18
  %%%, 17

  add_edge_color, 2
  add_edge_size, 3
  add_edge_zindex, 4
  add_igraph_info, 5
  add_listener, 5
  add_node_hidden, 6
  add_node_labels, 6
  add_node_size, 7

  convert_to_spring_weights, 8

  get_color_map, 9
  get_legend, 9

  highlight_multiple_connected, 10

  kgraph_to_lgraph, 10

  l_graph_to_igraph, 11
  lesMis, 11

  multiline_labels, 12

  renderSgraph, 13

  scale_graph, 13
  sgraph_clusters, 15
  sgraphOutput, 14
  sigma_from_igraph, 16