Package: ggsem (via r-universe)

December 21, 2024

Type Package

Title Interactively Visualize Structural Equation Modeling Diagrams

Version 0.1.2

Maintainer Seung Hyun Min <seung.min@mail.mcgill.ca>

Description It allows users to perform interactive and reproducible visualizations of path diagrams for structural equation modeling (SEM) and small-to-medium sized networks using the 'ggplot2' engine. Its 'shiny' app provides an interface that allows extensive customization, and creates CSV outputs, which can then be used to recreate the figures either using the 'shiny' app or in a typical 'ggplot2' workflow.

License GPL-2

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.4.0)

Imports ggplot2, igraph, shiny, DT, colourpicker, grid, svglite, grDevices, stats, utils, lavaan

Suggests semPlot

URL https://smin95.github.io/ggsem/

BugReports https://github.com/smin95/ggsem/issues/

Config/Needs/website rmarkdown

NeedsCompilation no

Author Seung Hyun Min [aut, cre]

Repository CRAN

Date/Publication 2024-12-21 03:20:02 UTC

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev libglpk-dev make libpng-dev libxml2-dev zlib1g-dev

Contents

`csv_to_ggplot` *Convert CSV files (from ggsem Shiny app) to ggplot output*

Description

This function converts the four CSV files from the ggsem Shiny app into a ggplot output object. The ggplot output can then be modified using standard ggplot2 functions, such as ggtitle() and annotate().

Usage

```
csv_to_ggplot(
  points_data = NULL,
  lines_data = NULL,
  annotations_data = NULL,
  loops_data = NULL,
  element_order = c("lines", "points", "self_loops", "annotations"),
  zoom_level = 1.2,
  horizontal_position = 0,
  vertical_position = 0,
  n = 100
)
```

Arguments

`points_data`	An object that stores the CSV file containing information about points from the ggsem shiny app. The default is NULL.
`lines_data`	An object that stores the CSV file containing information about lines from the ggsem shiny app. The default is NULL.
`annotations_data`	An object that stores the CSV file containing information about text annotations from the ggsem shiny app. The default is NULL.
`loops_data`	An object that stores the CSV file containing information about self-loop arrows from the ggsem shiny app. The default is NULL.

`element_order`	Order of the graphical elements on display. This is the order in which the graphical elements are added. So if it is written later, then it gets added later (more front), such as: c("lines", "points", "self_loops", "annotations"), which sets annotations to be added last (and hence most front).
`zoom_level`	A numeric value to control the zoom level of the plot. Default is 1.2.
`horizontal_position`	A numeric value for adjusting the horizontal position of the plot. Default is 0.
`vertical_position`	A numeric value for adjusting the vertical position of the plot. Default is 0.
`n`	Number of points to be used for interpolation (for gradient lines or curved lines). Default is 100.

Value

A ggplot object is returned as the function's output.

Examples

```
library(ggplot2)

# CSV files from ggsem app
points_data <- data.frame(
x = 5, y = 5, shape = 'square', color = '#D0C5ED', size = 50,
border_color = '#9646D4', border_width = 2, alpha = 1,
locked = FALSE, lavaan = FALSE
)

lines_data <- data.frame(
x_start = 2, y_start = -2, x_end = 8, y_end = -2, ctrl_x = NA, ctrl_y = NA,
type = 'Straight Line', color = '#000000', end_color = NA, color_type = 'Single',
gradient_position = NA, width = 1, alpha = 1, arrow = FALSE,
arrow_type = NA, arrow_size = NA, two_way = FALSE, lavaan = FALSE,
line_style = 'solid'
)

csv_to_ggplot(points_data = points_data,
              lines_data = lines_data,
              zoom_level = 1.4, # From the ggsem app
              horizontal_position = 14, # From the ggsem app
              element_order = c('lines', 'points')) # order priority: lines < points
```

`draw_annotations` *Write text annotations from an annotation CSV file (from ggsem Shiny app) on a ggplot object*

Description

This function adds text annotations onto any ggplot output (including your own plots not created from the ggsem Shiny app).

Usage

```
draw_annotations(p, annotations_data, zoom_level = 1)
```

Arguments

`p`	A ggplot2 object
`annotations_data`	The object that stores the CSV file containing information about text annotations from the ggsem Shiny app.
`zoom_level`	A numeric value to control the zoom level of the plot. Default is 1.

Value

A ggplot object is returned as the function's output.

Examples

```
library(ggplot2)

annotations_data <- data.frame(
text = 'Square One', x = 26, y = 300, font = 'serif',
size = 20, color = '#000000', angle = 0, alpha = 1,
fontface = 'bold', math_expression = FALSE,
lavaan = FALSE
)

p <- ggplot(mtcars) + geom_point(aes(mpg, disp))

draw_annotations(p, annotations_data, zoom_level = 1.2)
```

`draw_lines` *Draw lines from a line CSV File (from ggsem Shiny app) on a ggplot object*

Description

This function adds lines onto any ggplot output (including your own plots not created from the ggsem Shiny app).

Usage

```
draw_lines(p, lines_data, zoom_level = 1, n = 100)
```

Arguments

`p`	A ggplot object
`lines_data`	An object that stores the CSV file containing information about lines from the ggsem Shiny app.
`zoom_level`	A numeric value to control the zoom level of the plot. Default is 1.
`n`	Number of points to be used for interpolation (for gradient lines or curved lines). Default is 100.

Value

A ggplot object is returned as the function's output.

Examples

```
library(ggplot2)

lines_data <- data.frame(
x_start = 2, y_start = -2, x_end = 10, y_end = -2, ctrl_x = NA, ctrl_y = NA,
type = 'Straight Line', color = '#000000', end_color = '#cc3d3d', color_type = 'Gradient',
gradient_position = 0.35, width = 1.5, alpha = 1, arrow = FALSE,
arrow_type = NA, arrow_size = NA, two_way = FALSE, lavaan = FALSE,
line_style = 'solid'
)

p <- ggplot(mtcars) + geom_point(aes(mpg, disp))

draw_lines(p, lines_data, zoom_level = 1.2, n = 400)
```

`draw_loops` *Draw self-loop arrows from a self-loop arrow CSV file (from ggsem Shiny app) on a ggplot object*

Description

This function adds self-loop arrows onto any ggplot output (including your own plots not created from the ggsem shiny app).

Usage

```
draw_loops(p, loops_data, zoom_level = 1)
```

Arguments

`p`	A ggplot object
`loops_data`	An object that stores the CSV file containing information about self-loop arrows from the ggsem shiny app.
`zoom_level`	A numeric value to control the zoom level of the plot. Default is 1.

Value

A ggplot object is returned as the function's output.

Examples

```
library(ggplot2)

loops_data <- data.frame(
x_center = -5, y_center = 5, radius = 2, color = '#000000', width = 1,
alpha = 1, arrow_type = 'closed', arrow_size = 0.1, gap_size = 0.2,
loop_width = 1, loop_height = 20, orientation = 0,
two_way = FALSE, locked = FALSE
)

p <- ggplot(mtcars) + geom_point(aes(mpg, disp))

draw_loops(p, loops_data, zoom_level = 1.2)
```

draw_points   *Draw points from a point CSV file (from ggsem Shiny app) on a ggplot object*

Description

This function adds points onto any ggplot output (including your own plots not created from the ggsem shiny app).

Usage

```
draw_points(p, points_data, zoom_level = 1)
```

Arguments

`p`	A ggplot object
`points_data`	An object that stores the CSV file containing information about points from the ggsem shiny app.
`zoom_level`	A numeric value to control the zoom level of the plot. Default is 1.

Value

A ggplot object is returned as the function's output.

Examples

```
library(ggplot2)

points_data <- data.frame(
x = 20, y = 300, shape = 'square', color = '#D0C5ED', size = 50,
border_color = '#9646D4', border_width = 2, alpha = 1,
locked = FALSE, lavaan = FALSE
)

p <- ggplot(mtcars) + geom_point(aes(mpg, disp))

draw_points(p, points_data, zoom_level = 1.2)
```

`get_axis_range` *Get axis range of a ggplot object*

Description

A function to calculate the range of x- and y- axes.

Usage

```
get_axis_range(plot)
```

Arguments

`plot`	ggplot output from csv_to_ggplot()

Value

A list object that has two elements, each of which has two vector values. The first element stores the minimum and maximum values of the plot's x-axis range, while the second element stores the minimum and maximum values of the plot's y-axis range.

Examples

```
library(ggplot2)
ggplot(mtcars) + geom_point(aes(mpg, disp)) -> p1
get_axis_range(p1)
```

`ggsem` *Run ggsem (shiny app) locally through a browser*

Description

Run ggsem (shiny app) locally through a browser

Usage

```
ggsem()
```

Value

No return value, called for side effects

Index