Package ‘ggsem’

February 2, 2025

Type Package

Title Interactively Visualize Structural Equation Modeling Diagrams

Version 0.2.4

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

Description
It is an R package and web-based application, allowing users to perform interactive and reproducible visualizations of path diagrams for structural equation modeling (SEM) and networks using the 'ggplot2' engine. Its app (built with 'shiny') provides an interface that allows extensive customization, and creates CSV outputs, which can then be used to recreate the figures either using the web app or script-based 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, smplot2, utils, lavaan, Rtsne, umap

Suggests semPlot, dplyr

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 2025-02-02 22:50:02 UTC

Contents

`adjust_axis_range` *Adjust Axis Range of a Plot of a ggplot2 Plot*

Description

This function modifies the axis ranges of a ggplot object, with optional user-specified ranges, additional buffers, and the ability to enforce a fixed aspect ratio. This is a modified version of `adjust_axis_space()`.

Usage

```
adjust_axis_range(
  plot,
  x_range = NULL,
  y_range = NULL,
  buffer_percent = 0,
  fixed_aspect_ratio = TRUE
)
```

Arguments

`plot`	A ggplot object. The plot whose axis ranges are to be adjusted.
`x_range`	A numeric vector of length 2 specifying the desired x-axis range. If `NULL`, the current x-axis range is retained. Default is `NULL`.
`y_range`	A numeric vector of length 2 specifying the desired y-axis range. If `NULL`, the current y-axis range is retained. Default is `NULL`.
`buffer_percent`	A numeric value indicating the percentage of additional space to add to each axis range as a buffer. Default is ‘0‘ (no buffer).
`fixed_aspect_ratio`	A logical value indicating whether to maintain a fixed aspect ratio for the plot. If `TRUE`, the function adjusts one axis to preserve the aspect ratio. Default is `TRUE`.

Details

- If ‘x_range‘ or ‘y_range‘ are provided, these values will override the current axis ranges. - The ‘buffer_percent‘ parameter adds proportional space to the axis ranges, calculated as a percentage of the range’s width or height. - When ‘fixed_aspect_ratio‘ is ‘TRUE‘, the function adjusts either the x-axis or y-axis to ensure the plot maintains a fixed aspect ratio.

Value

A modified ggplot object with adjusted axis ranges.

Examples

```
# CSV files from ggsem app
points_data <- data.frame(
x = 20, y = 20, shape = 'rectangle', color = '#D0C5ED', size = 50,
border_color = '#9646D4', border_width = 2, alpha = 1, width_height_ratio = 1.6, orientation = 45,
lavaan = FALSE, lavaan = FALSE, network = FALSE, locked = FALSE
)

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,
network = FALSE, line_style = 'solid', locked = FALSE
)

p <- csv_to_ggplot(points_data = points_data,
              lines_data = lines_data,
              zoom_level = 1.2, # Value from the ggsem app
              horizontal_position = 0, # Value from the ggsem app
              element_order = c('lines', 'points')) # order priority: lines < points


adjust_axis_range(p, x_range = c(-30,30), y_range= c(-30,30))
```

`adjust_axis_space` *Adjust Surrounding White Space of a ggplot2 Plot*

Description

This function allows users to remove or manage whitespace around graphical elements. It supports asymmetrical adjustments for each boundary (left, right, bottom, and top). Users can also maintain a fixed aspect ratio if required.

Usage

```
adjust_axis_space(
  plot,
  x_adjust_left_percent = 0,
  x_adjust_right_percent = 0,
  y_adjust_bottom_percent = 0,
  y_adjust_top_percent = 0,
  fixed_aspect_ratio = TRUE
)
```

Arguments

`plot`	A ggplot2 object. The plot whose axis ranges need adjustment.
`x_adjust_left_percent`	Numeric. Percentage by which to expand the left boundary of the x-axis. Default is `0`.
`x_adjust_right_percent`	Numeric. Percentage by which to expand the right boundary of the x-axis. Default is `0`.
`y_adjust_bottom_percent`	Numeric. Percentage by which to expand the bottom boundary of the y-axis. Default is `0`.
`y_adjust_top_percent`	Numeric. Percentage by which to expand the top boundary of the y-axis. Default is `0`‘.
`fixed_aspect_ratio`	Logical. If `TRUE`, maintains a fixed aspect ratio (1:1). If ‘FALSE‘, allows independent scaling for x and y axes. Default is `TRUE`.

Details

- **Percentage Adjustments:** The percentages provided for each axis boundary are calculated based on the current axis range. For example, `x_adjust_left_percent = 10` expands the left boundary by 10 - **Fixed Aspect Ratio:** When `fixed_aspect_ratio = TRUE`, the function adjusts either the x-axis or y-axis to maintain a 1:1 aspect ratio. The larger adjustment determines the scaling for both axes.

Value

A ggplot2 object with adjusted axis ranges. The adjusted plot retains its original attributes and is compatible with additional ggplot2 layers and themes.

Examples

```
# CSV files from ggsem app
points_data <- data.frame(
x = 20, y = 20, shape = 'rectangle', color = '#D0C5ED', size = 50,
border_color = '#9646D4', border_width = 2, alpha = 1, width_height_ratio = 1.6, orientation = 45,
lavaan = FALSE, lavaan = FALSE, network = FALSE, locked = FALSE
)

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,
network = FALSE, line_style = 'solid', locked = FALSE
)

p <- csv_to_ggplot(points_data = points_data,
              lines_data = lines_data,
              zoom_level = 1.2, # Value from the ggsem app
              horizontal_position = 0, # Value from the ggsem app
              element_order = c('lines', 'points')) # order priority: lines < points


adjust_axis_space(p, x_adjust_left_percent = 10, x_adjust_right_percent = 10,
            y_adjust_bottom_percent = 5, y_adjust_top_percent = 5)
```

csv_to_ggplot *Convert CSV Files (from ggsem Shiny App) to a ggplot Object*

Description

This function converts the CSV files exported from the ggsem Shiny app into a customizable ggplot object. The resulting plot is compatible with ggplot2 functions, allowing users to modify it further (e.g., adding titles or annotations).

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`	A data frame containing point data exported from the ggsem Shiny app. Default is `NULL`.
`lines_data`	A data frame containing line data exported from the ggsem Shiny app. Default is ‘NULL‘.
`annotations_data`	A data frame containing text annotation data exported from the ggsem Shiny app. Default is `NULL`.
`loops_data`	A data frame containing self-loop arrow data exported from the ggsem Shiny app. Default is `NULL`.
`element_order`	A character vector specifying the order in which graphical elements are added to the plot. For example: `c("lines", "points", "self_loops", "annotations")`. Later elements appear on top. Default includes all elements.

`zoom_level`	A numeric value controlling the zoom level of the plot. A value >1 zooms in; <1 zooms out. Default is `1.2`.
`horizontal_position`	A numeric value to shift the plot horizontally. Default is `0`.
`vertical_position`	A numeric value to shift the plot vertically. Default is `0`.
`n`	Number of points used for interpolation in gradient or curved lines. Default is `100`.

Details

- The function uses ‘coord_fixed‘ to ensure square plotting space and uniform scaling. - The ‘element_order‘ parameter determines the layering of graphical elements, with later elements appearing on top. - The ‘axis_ranges‘ attribute is attached to the plot for additional programmatic access.

Value

A ggplot object with an `axis_ranges` attribute specifying the x and y axis ranges after adjustments.

Examples

```
# CSV files from ggsem app
points_data <- data.frame(
x = 20, y = 20, shape = 'rectangle', color = '#D0C5ED', size = 50,
border_color = '#9646D4', border_width = 2, alpha = 1, width_height_ratio = 1.6, orientation = 45,
lavaan = FALSE, lavaan = FALSE, network = FALSE, locked = FALSE
)

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,
network = FALSE, line_style = 'solid', locked = FALSE
)

csv_to_ggplot(points_data = points_data,
              lines_data = lines_data,
              zoom_level = 1.2, # Value from the ggsem app
              horizontal_position = 0, # Value from the ggsem app
              element_order = c('lines', 'points')) # order priority: lines < points
```

`draw_annotations` *Draw Text Annotations to a ggplot Object*

Description

This function overlays text annotations onto any ggplot object. It is particularly useful for adding annotations from CSV files generated by the ggsem Shiny app but can also be used with custom annotation data.

Usage

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

Arguments

`p`	A ggplot object. The plot to which the annotations will be added.
`annotations_data`	A data frame containing annotation information. Typically, this comes from a CSV file generated by the ggsem Shiny app. The required columns include: • `text`: The text to annotate (character). • `x`, `y`: The coordinates for the text (numeric). • `font`: The font family to use (character, e.g., "serif"). • `size`: The size of the text (numeric). • `color`: The color of the text (character, valid hex color). • `angle`: The rotation angle of the text (numeric, in degrees). • `alpha`: The transparency of the text (numeric, 0 to 1). • `fontface`: The font style (character, e.g., "bold"). • `math_expression`: Logical, whether the text should be parsed as a mathematical expression.
`zoom_level`	Numeric. Adjusts the size of annotations based on the zoom level. Default is `1`.

Value

A ggplot object with the specified annotations added.

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, network = FALSE, locked = FALSE
)
```

```
p <- ggplot()

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

`draw_lines` *Draw Lines on a ggplot Object from Line Data*

Description

This function overlays lines or arrows to a ggplot object based on line data. It supports straight lines, curved lines, gradient color transitions, and one-way or two-way arrows. The data can come from a CSV file generated by the ggsem Shiny app or custom input.

Usage

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

Arguments

`p`	A ggplot object to which the lines will be added.
`lines_data`	A data frame containing line information. The expected columns include: • `x_start`, `y_start`: Starting coordinates of the line. • `x_end`, `y_end`: Ending coordinates of the line. • `ctrl_x`, `ctrl_y`: Control points for curved lines (optional). • `type`: Type of line (`"Straight Line"`, `"Curved Line"`, `"Straight Arrow"`, or `"Curved Arrow"`). • `color`: Start color of the line (hexadecimal color code). • `end_color`: End color of the line for gradients (hexadecimal color code). • `color_type`: `"Gradient"` for gradient lines, or `"Single"` for solid-colored lines. • `gradient_position`: Position of the gradient transition along the line (numeric, 0 to 1). • `width`: Width of the line (numeric). • `alpha`: Transparency of the line (numeric, 0 to 1). • `arrow`: Logical, whether an arrowhead is used. • `arrow_type`: Type of arrow (`"open"`, `"closed"`, etc.). • `arrow_size`: Size of the arrowhead. • `two_way`: Logical, whether the line has two arrowheads (bidirectional). • `line_style`: Line style (`"solid"`, `"dashed"`, or `"dotted"`).
`zoom_level`	Numeric. Adjusts the size of line widths and arrowheads relative to the plot. Default is `1`.
`n`	Integer. Number of points for interpolation in gradient or curved lines. Default is `100`.

Value

A ggplot object with the specified lines or arrows added.

Examples

```
library(ggplot2)

lines_df <- data.frame(
x_start = 2, y_start = -2, x_end = 6, 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,
network = FALSE, line_style = 'solid', locked = FALSE
)

p <- ggplot()

draw_lines(p, lines_data = lines_df, zoom_level = 1.2, n = 200)
```

`draw_loops` *Draw Self-loop Arrows on a ggplot Object*

Description

This function overlays self-loop arrows to a ggplot object based on data describing their positions, sizes, orientations, and styles. Self-loop arrows can be drawn in one direction or bidirectionally with customizable parameters such as color, width, and arrow type. The data can come from a CSV file generated by the ggsem Shiny app or custom input.

Usage

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

Arguments

`p`	A ggplot object to which the self-loop arrows will be added.
`loops_data`	A data frame containing information about the self-loop arrows. The expected columns include:

- `x_center`, `y_center`: Center coordinates of the loop.
- `radius`: Radius of the loop.
- `color`: Color of the loop (hexadecimal color code).
- `width`: Width of the loop line (numeric).
- `alpha`: Transparency of the loop line (numeric, 0 to 1).
- `arrow_type`: Type of arrow ("`closed`" or "`open`").
- `arrow_size`: Size of the arrowhead.

- `gap_size`: Size of the gap in the loop, specified as a fraction of the full circle (numeric, 0 to 1).
- `loop_width`, `loop_height`: Width and height scaling factors for the loop.
- `orientation`: Rotation angle of the loop in degrees.
- `two_way`: Logical, whether the loop is bidirectional (adds arrows at both ends).

`zoom_level`	Numeric. Adjusts the size of line widths and arrowheads relative to the plot. Default is `1`.

Value

A ggplot object with the specified self-loop arrows added.

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 = 5, loop_height = 5, orientation = 0,
two_way = FALSE, locked = FALSE
)

p <- ggplot()

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

`draw_points` *Draw Points on a ggplot Object*

Description

This function overlays points to a ggplot object using data from a CSV file generated by the ggsem Shiny app or any custom dataset. Points can be styled with various shapes, colors, sizes, and orientations.

Usage

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

Arguments

`p`	A ggplot object to which the points will be added.
`points_data`	A data frame containing information about the points to be drawn. The expected columns include:

- `x`, `y`: Coordinates of the point.

- `shape`: Shape of the point (`"circle"`, `"square"`, `"triangle"`, `"rectangle"`, `"oval"`, or `"diamond"`).
- `color`: Fill color of the point (hexadecimal color code).
- `size`: Size of the point.
- `border_color`: Border color of the point (hexadecimal color code).
- `border_width`: Width of the border.
- `alpha`: Transparency of the point (numeric, 0 to 1).
- `width_height_ratio`: Ratio of width to height (for shapes like rectangles and ovals).
- `orientation`: Rotation angle of the point in degrees (for shapes like rectangles and diamonds).

`zoom_level`	Numeric. Adjusts the size of the points relative to the plot. Default is `1`.

Value

A ggplot object with the specified points added.

Examples

```
library(ggplot2)

points_data <- data.frame(
x = 20, y = 20, shape = 'rectangle', color = '#D0C5ED', size = 50,
border_color = '#9646D4', border_width = 2, alpha = 1, width_height_ratio = 1.6, orientation = 45,
lavaan = FALSE, lavaan = FALSE, network = FALSE, locked = FALSE
)

p <- ggplot()

draw_points(p, points_data, zoom_level = 1.2) +
scale_x_continuous(limits = c(0,50)) +
scale_y_continuous(limits = c(0,50))
```

`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

`save_figure` *Save a ggplot Object with Adjusted Dimensions*

Description

This function saves a ggplot object (created from 'csv_to_ggplot()' function) to a file with dimensions automatically determined based on the x-axis and y-axis ranges of the plot. The size of the output can be further controlled using addtional arguments.

Usage

```
save_figure(
  filename,
  plot,
  units = "in",
  dpi = 300,
  aspect_ratio = NULL,
  scale_factor = 0.11,
  ...
)
```

Arguments

`filename`	A string. The name of the output file (e.g., "plot.png").
`plot`	A ggplot object to save.
`units`	A string. Units for width and height. Default is `"in"` (inches). Other options include `"cm"` or `"mm"`.
`dpi`	Numeric. Resolution of the output file in dots per inch. Default is 300.
`aspect_ratio`	Numeric or `NULL`. If provided, fixes the aspect ratio of the plot (e.g., `1` for square). If `NULL`, uses the natural data aspect ratio. Default is `NULL`.
`scale_factor`	Numeric. A scaling factor to control the overall size of the saved plot. Default is `0.11`.
`...`	Additional arguments passed to `ggsave()`.

Value

Saves the ggplot object to the specified file and does not return a value.

Examples

```
## Not run:
# CSV files from ggsem app
points_data <- data.frame(
x = 20, y = 20, shape = 'rectangle', color = '#D0C5ED', size = 50,
border_color = '#9646D4', border_width = 2, alpha = 1, width_height_ratio = 1.6, orientation = 45,
lavaan = FALSE, lavaan = FALSE, network = FALSE, locked = FALSE
)

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,
network = FALSE, line_style = 'solid', locked = FALSE
)

p <- csv_to_ggplot(points_data = points_data,
              lines_data = lines_data,
              zoom_level = 1.2, # Value from the ggsem app
              horizontal_position = 0, # Value from the ggsem app
              element_order = c('lines', 'points')) # order priority: lines < points


p1 <- adjust_axis_range(p, x_adjust_left_percent = 10, x_adjust_right_percent = 10,
             y_adjust_bottom_percent = 5, y_adjust_top_percent = 5)


# Save with default scaling
save_figure("p1.png", p1)

## End(Not run)
```

Index