Package: plotthis (via r-universe)

November 21, 2024

Title High-Level Plotting Built Upon 'ggplot2' and Other Plotting Packages

Version 0.3.6

Description Provides high-level API and a wide range of options to create stunning, publication-quality plots effortlessly. It is built upon 'ggplot2' and other plotting packages, and is designed to be easy to use and to work seamlessly with 'ggplot2' objects. It is particularly useful for creating complex plots with multiple layers, facets, and annotations. It also provides a set of functions to create plots for specific types of data, such as Venn diagrams, alluvial diagrams, and phylogenetic trees. The package is designed to be flexible and customizable, and to work well with the 'ggplot2' ecosystem. The API can be found at <https://pwwang.github.io/plotthis/reference/index.html>.

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/pwwang/plotthis

https://pwwang.github.io/plotthis/

BugReports https://github.com/pwwang/plotthis/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.0.0)

Imports circlize, ggplot2, rlang, dplyr, tidyr, glue, forcats, gtable, reshape2, stringr, scales, gridtext, methods, patchwork, ggrepel, ggnewscale, cowplot, zoo

Suggests testthat, ComplexHeatmap, clustree, gglogger, ggwordcloud, ggalluvial, ggVennDiagram (>= 1.5.0), ggupset, ggpubr, ggforce, ggraph, ggridges, hexbin, igraph, iNEXT, scattermore, concaveman

LazyData true

Config/Needs/website rmarkdown

Config/pak/sysreqs libicu-dev libjpeg-dev libpng-dev libxml2-dev libssl-dev

Repository https://pwwang.r-universe.dev

RemoteUrl https://github.com/pwwang/plotthis

RemoteRef HEAD

RemoteSha fb026b60c805838f0f6f0b5d4e3c58434e68d68c

Contents

`AreaPlot` *Area plot*

Description

A plot showing how one or more groups' numeric values change over the progression of a another variable

Usage

```
AreaPlot(
  data,
  x,
  y = NULL,
  x_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  scale_y = FALSE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
```

```
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to name the legend of fill.
scale_y	A logical value to scale the y-axis by the total number in each x-axis group.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
x_text_angle	A numeric value specifying the angle of the x-axis text.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

keep_empty A logical value indicating whether to keep empty groups. If FALSE, empty groups will be removed.

seed The random seed to use. Default is 8525.

combine Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow A numeric value specifying the number of rows in the facet.

ncol A numeric value specifying the number of columns in the facet.

byrow A logical value indicating whether to fill the plots by row.

... Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- data.frame(
    x = rep(c("A", "B", "C", "D"), 2),
    y = c(1, 3, 6, 4, 2, 5, 7, 8),
    group = rep(c("F1", "F2"), each = 4)
)
AreaPlot(data, x = "x", y = "y", group_by = "group")
AreaPlot(data, x = "x", y = "y", group_by = "group",
         scale_y = TRUE)
AreaPlot(data, x = "x", y = "y", split_by = "group")
AreaPlot(data, x = "x", y = "y", split_by = "group", palette = c(F1 = "Blues", F2 = "Reds"))
```

BarPlot *Bar Plot*

Description

- BarPlot is used to create a bar plot.
- SplitBarPlot (a.k.a WaterfallPlot) is used to create a bar plot with splitting the bars on the two sides.

Usage

```
BarPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  flip = FALSE,
  fill_by_x_if_no_group = TRUE,
  line_name = NULL,
  label_nudge = 0,
  label = NULL,
  label_fg = "black",
  label_size = 4,
  label_bg = "white",
  label_bg_r = 0.1,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  facet_args = list(),
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_line = NULL,
  line_color = "red2",
  line_width = 0.6,
  line_type = 2,
  add_trend = FALSE,
  trend_color = "black",
  trend_linewidth = 1,
```

```
  trend_ptsize = 2,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  x_text_angle = 0,
  aspect.ratio = 1,
  y_min = NULL,
  y_max = NULL,
  position = "auto",
  position_dodge_preserve = "total",
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  expand = waiver(),
  width = waiver(),
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)

SplitBarPlot(
  data,
  x,
  y,
  y_sep = "_",
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  alpha_by = NULL,
  alpha_reverse = FALSE,
  alpha_name = NULL,
  order_y = list(`+` = c("x_desc", "alpha_desc"), `-` = c("x_desc", "alpha_asc")),
  bar_height = 0.9,
  lineheight = 0.5,
  max_charwidth = 80,
  fill_by = NULL,
  fill_by_sep = "_",
  fill_name = NULL,
  direction_pos_name = "positive",
```

```
  direction_neg_name = "negative",
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  facet_by = NULL,
  facet_scales = "free_y",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  x_min = NULL,
  x_max = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)

WaterfallPlot(
  data,
  x,
  y,
  y_sep = "_",
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  alpha_by = NULL,
  alpha_reverse = FALSE,
  alpha_name = NULL,
 order_y = list(`+` = c("x_desc", "alpha_desc"), `-` = c("x_desc", "alpha_asc")),
  bar_height = 0.9,
  lineheight = 0.5,
  max_charwidth = 80,
  fill_by = NULL,
  fill_by_sep = "_",
  fill_name = NULL,
  direction_pos_name = "positive",
  direction_neg_name = "negative",
```

```
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  facet_by = NULL,
  facet_scales = "free_y",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  x_min = NULL,
  x_max = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided.
y	A character string specifying the column name of the data frame to plot for the y-axis.
flip	A logical value indicating whether to flip the x and y axes.
fill_by_x_if_no_group	A logical value indicating whether to fill the bars by the x-axis values if there is no group_by.
line_name	A character string indicating the name of the line.
label_nudge	A numeric value to nudge the labels (the distance between the label and the top of the bar).
label	A column name for the values to be displayed on the top of the bars. If TRUE, the y values will be displayed.
label_fg	A character string indicating the color of the label.

label_size	A numeric value indicating the size of the label.
label_bg	A character string indicating the background color of the label.
label_bg_r	A numeric value indicating the radius of the background.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to specify the name of the group_by in the legend.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
facet_args	A list of arguments to pass to ggplot2::facet_grid or ggplot2::facet_wrap.
add_bg	A logical value indicating whether to add a background to the plot.
bg_palette	A character string indicating the palette to use for the background.
bg_palcolor	A character string indicating the color to use for the background.
bg_alpha	A numeric value indicating the alpha of the background.
add_line	A numeric value indicating the y value to add a horizontal line.
line_color	A character string indicating the color of the line.
line_width	A numeric value indicating the size of the line.
line_type	A numeric value indicating the type of the line.
add_trend	A logical value to add trend line to the plot.
trend_color	A character string to specify the color of the trend line.
trend_linewidth	A numeric value to specify the width of the trend line.
trend_ptsize	A numeric value to specify the size of the trend line points.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

alpha
: A numeric value specifying the transparency of the plot.

x_text_angle
: A numeric value specifying the angle of the x-axis text.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

y_min
: A numeric value to specify the minimum value of the y axis.

y_max
: A numeric value to specify the maximum value of the y axis.

position
: A character string indicating the position of the bars. If "auto", the position will be "stack" if group_by has more than 5 levels, otherwise "dodge". "fill" is also a valid option. Only works when group_by is not NULL.

position_dodge_preserve
: Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

keep_empty
: A logical value indicating whether to keep empty groups. If FALSE, empty groups will be removed.

expand
: The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html

width
: A numeric value specifying the width of the bars.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow	A logical value indicating whether to fill the plots by row.
seed	The random seed to use. Default is 8525.
...	Additional arguments.
y_sep	A character string to concatenate the x columns if there are multiple.
alpha_by	A character string indicating the column name to use for the transparency of the bars.
alpha_reverse	A logical value indicating whether to reverse the transparency.
alpha_name	A character string indicating the legend name of the transparency.
order_y	A list of character strings indicating the order of the y axis. The keys are "+", "-", or "". *However, "+/-" should not be mixed with ""*. The values are "x_asc", "x_desc", "alpha_asc", or "alpha_desc", indicating how to order the y axis. The default is `list("+" = c("x_desc", "alpha_desc"), "-" = c("x_desc", "alpha_asc"))`, meaning the positive values are ordered by the x-axis values in descending order and the alpha values in descending order, and the negative values are ordered by the x-axis values in descending order and the alpha values in ascending order. The "*" key is used to order the y axis without considering the direction.
bar_height	A numeric value indicating the height of the bars.
lineheight	A numeric value indicating the height of the text.
max_charwidth	A numeric value indicating the maximum width of the text.
fill_by	A character string indicating the column name to use for the fill of the bars.
fill_by_sep	A character string to concatenate the fill columns if there are multiple.
fill_name	A character string indicating the legend name of the fill.
direction_pos_name	A character string indicating the name of the positive direction.
direction_neg_name	A character string indicating the name of the negative direction.
x_min	A numeric value indicating the minimum value of the x axis.
x_max	A numeric value indicating the maximum value of the x axis.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- data.frame(
    x = c("A", "B", "C", "D", "E", "F", "G", "H"),
    y = c(10, 8, 16, 4, 6, 12, 14, 2),
    group = c("G1", "G1", "G2", "G2", "G3", "G3", "G4", "G4"),
    facet = c("F1", "F2", "F3", "F4", "F1", "F2", "F3", "F4")
)

BarPlot(data, x = "x", y = "y")
BarPlot(data, x = "x", y = "y", fill_by_x_if_no_group = FALSE)
```

```
BarPlot(data, x = "x", y = "y", label = TRUE)
BarPlot(data, x = "x", y = "y", label = "facet", label_nudge = 1)
BarPlot(data, x = "group", y = "y", group_by = "x")
BarPlot(data,
    x = "group", y = "y", group_by = "x",
    position = "dodge", add_bg = TRUE
)
BarPlot(data,
    x = "x", y = "y", split_by = "group",
    facet_by = "facet", position = "dodge", facet_ncol = 1
)
BarPlot(data,
    x = "x", y = "y", split_by = "group",
    palette = list(G1 = "Reds", G2 = "Blues", G3 = "Greens", G4 = "Purp"),
    facet_by = "facet", position = "dodge", facet_ncol = 1
)
BarPlot(data,
    x = "group", y = "y", group_by = "x",
    position = "dodge", add_bg = TRUE, bg_palette = "Spectral"
)
# use the count
BarPlot(data, x = "group", ylab = "count")
# flip the plot
BarPlot(data, x = "group", flip = TRUE, ylab = "count")
data <- data.frame(
    word = c("apple", "banana", "cherry", "date", "elderberry",
             "It is a very long term with a lot of words"),
    count = c(-10, 20, -30, 40, 50, 34),
    score = c(1, 2, 3, 4, 5, 3.2),
    group = c("A", "A", "B", "B", "C", "C")
)
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score")
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
             max_charwidth = 30, lineheight = 1.1)
SplitBarPlot(data, x = "count", y = "word", fill_by = "group")
SplitBarPlot(data, x = "count", y = "word", facet_by = "group",
             fill_name = "Direction")
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score", split_by="group",
             palette = c(A = "Reds", B = "Blues", C = "Greens"))
```

BoxPlot *Box / Violin Plot*

Description

Box plot or violin plot with optional jitter points, trend line, statistical test, background, line, and highlight.

Usage

```
BoxPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  sort_x = c("none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc",
    "median"),
  flip = FALSE,
  keep_empty = FALSE,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  x_text_angle = ifelse(isTRUE(flip) && isTRUE(stack), 90, 45),
  fill_mode = ifelse(!is.null(group_by), "dodge", "x"),
  fill_reverse = FALSE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  add_point = FALSE,
  pt_color = "grey30",
  pt_size = NULL,
  pt_alpha = 1,
  jitter_width = 0.5,
  jitter_height = 0.1,
  stack = FALSE,
  y_max = NULL,
  y_min = NULL,
  add_trend = FALSE,
  trend_color = NULL,
  trend_linewidth = 1,
  trend_ptsize = 2,
  add_stat = NULL,
  stat_name = NULL,
  stat_color = "black",
  stat_size = 1,
  stat_stroke = 1,
  stat_shape = 25,
  add_bg = FALSE,
  bg_palette = "stripe",
```

```
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_line = NULL,
  line_color = "red2",
  line_width = 0.6,
  line_type = 2,
  highlight = NULL,
  highlight_color = "red2",
  highlight_size = 1,
  highlight_alpha = 1,
  comparisons = NULL,
  ref_group = NULL,
  pairwise_method = "wilcox.test",
  multiplegroup_comparisons = FALSE,
  multiple_method = "kruskal.test",
  sig_label = c("p.signif", "p.format"),
  sig_labelsize = 3.5,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)

ViolinPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
 sort_x = c("none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc",
    "median"),
  flip = FALSE,
  keep_empty = FALSE,
  group_by = NULL,
  group_by_sep = "_",
```

```
group_name = NULL,
x_text_angle = ifelse(isTRUE(flip) && isTRUE(stack), 90, 45),
fill_mode = ifelse(!is.null(group_by), "dodge", "x"),
fill_reverse = FALSE,
theme = "theme_this",
theme_args = list(),
palette = "Paired",
palcolor = NULL,
alpha = 1,
aspect.ratio = NULL,
legend.position = "right",
legend.direction = "vertical",
add_point = FALSE,
pt_color = "grey30",
pt_size = NULL,
pt_alpha = 1,
jitter_width = 0.5,
jitter_height = 0.1,
stack = FALSE,
y_max = NULL,
y_min = NULL,
add_box = FALSE,
box_color = "black",
box_width = 0.1,
box_ptsize = 2.5,
add_trend = FALSE,
trend_color = NULL,
trend_linewidth = 1,
trend_ptsize = 2,
add_stat = NULL,
stat_name = NULL,
stat_color = "black",
stat_size = 1,
stat_stroke = 1,
stat_shape = 25,
add_bg = FALSE,
bg_palette = "stripe",
bg_palcolor = NULL,
bg_alpha = 0.2,
add_line = NULL,
line_color = "red2",
line_width = 0.6,
line_type = 2,
highlight = NULL,
highlight_color = "red2",
highlight_size = 1,
highlight_alpha = 1,
comparisons = NULL,
```

```
  ref_group = NULL,
  pairwise_method = "wilcox.test",
  multiplegroup_comparisons = FALSE,
  multiple_method = "kruskal.test",
  sig_label = c("p.signif", "p.format"),
  sig_labelsize = 3.5,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided. When in_form is "wide", x columns will not be concatenated.
y	A character string specifying the column name of the data frame to plot for the y-axis.
in_form	A character string to specify the input data type. Either "long" or "wide".
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
sort_x	A character string to specify the sorting of x-axis. Either "none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc", "median".
flip	A logical value to flip the plot.
keep_empty	A logical value indicating whether to keep empty groups. If FALSE, empty groups will be removed.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to name the legend of dodge.

x_text_angle	A numeric value specifying the angle of the x-axis text.
fill_mode	A character string to specify the fill mode. Either "dodge", "x", "mean", "median".
fill_reverse	A logical value to reverse the fill colors for gradient fill (mean/median).
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
add_point	A logical value to add (jitter) points to the plot.
pt_color	A character string to specify the color of the points.
pt_size	A numeric value to specify the size of the points.
pt_alpha	A numeric value to specify the transparency of the points.
jitter_width	A numeric value to specify the width of the jitter.
jitter_height	A numeric value to specify the height of the jitter.
stack	A logical value whether to stack the facetted plot by 'facet_by'.
y_max	A numeric value or a character string to specify the maximum value of the y-axis.
y_min	A numeric value or a character string to specify the minimum value of the y-axis.
add_trend	A logical value to add trend line to the plot.
trend_color	A character string to specify the color of the trend line.
trend_linewidth	A numeric value to specify the width of the trend line.
trend_ptsize	A numeric value to specify the size of the trend line points.
add_stat	A character string to add statistical test to the plot.
stat_name	A character string to specify the name of the stat legend.
stat_color	A character string to specify the color of the statistical test.
stat_size	A numeric value to specify the size of the statistical test.
stat_stroke	A numeric value to specify the stroke of the statistical test.

stat_shape A numeric value to specify the shape of the statistical test.

add_bg A logical value to add background to the plot.

bg_palette A character string to specify the palette of the background.

bg_palcolor A character vector to specify the colors of the background.

bg_alpha A numeric value to specify the transparency of the background.

add_line A character string to add a line to the plot.

line_color A character string to specify the color of the line.

line_width A numeric value to specify the size of the line.

line_type A numeric value to specify the type of the line.

highlight A vector of character strings to highlight the points. It should be a subset of the row names of the data. If TRUE, it will highlight all points.

highlight_color A character string to specify the color of the highlighted points.

highlight_size A numeric value to specify the size of the highlighted points.

highlight_alpha A numeric value to specify the transparency of the highlighted points.

comparisons A logical value or a list of vectors to perform pairwise comparisons.

ref_group A character string to specify the reference group for comparisons.

pairwise_method A character string to specify the pairwise comparison method.

multiplegroup_comparisons A logical value to perform multiple group comparisons.

multiple_method A character string to specify the multiple group comparison method.

sig_label A character string to specify the label of the significance test.

sig_labelsize A numeric value to specify the size of the significance test label.

facet_by A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow A logical value indicating whether to fill the plots by row. Default is TRUE.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab	A character string specifying the y-axis label.
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.
add_box	A logical value to add box plot to the plot.
box_color	A character string to specify the color of the box plot.
box_width	A numeric value to specify the width of the box plot.
box_ptsize	A numeric value to specify the size of the box plot points in the middle.

Value

The Box / Violin plot(s). When split_by is not provided, it returns a ggplot object. When split_by is provided, it returns a object of plots wrapped by patchwork::wrap_plots if combine = TRUE; otherwise, it returns a list of ggplot objects.

Examples

```
set.seed(8525)
data <- data.frame(
    x = rep(LETTERS[1:8], 40),
    y = rnorm(320),
    group1 = sample(c("g1", "g2"), 320, replace = TRUE),
    group2 = sample(c("h1", "h2", "h3", "h4"), 320, replace = TRUE)
)

BoxPlot(data, x = "x", y = "y")
BoxPlot(data,
    x = "x", y = "y",
    stack = TRUE, flip = TRUE, facet_by = "group1",
    add_bg = TRUE, bg_palette = "Paired"
)
BoxPlot(data,
    x = "x", y = "y",
    stack = TRUE, flip = TRUE, split_by = "group1",
    add_bg = TRUE, bg_palette = "Paired",
    palcolor = list(g1 = c("red", "blue"), g2 = c("blue", "red"))
)

# wide form data
data_wide <- data.frame(
    A = rnorm(100),
    B = rnorm(100),
    C = rnorm(100)
)
BoxPlot(data_wide, x = c("A", "B", "C"), in_form = "wide")
```

```
ViolinPlot(data, x = "x", y = "y")
ViolinPlot(data, x = "x", y = "y", add_box = TRUE)
ViolinPlot(data, x = "x", y = "y", add_point = TRUE)
ViolinPlot(data, x = "x", y = "y", add_trend = TRUE)
ViolinPlot(data, x = "x", y = "y", add_stat = mean)
ViolinPlot(data, x = "x", y = "y", add_bg = TRUE)
ViolinPlot(data, x = "x", y = "y", add_line = 0)
ViolinPlot(data, x = "x", y = "y", group_by = "group1")
ViolinPlot(data,
    x = "x", y = "y", group_by = "group1",
    facet_by = "group2", add_box = TRUE
)
ViolinPlot(data, x = "x", y = "y", add_point = TRUE, highlight = 'group1 == "g1"',
    alpha = 0.8, highlight_size = 1.5, pt_size = 1, add_box = TRUE)
ViolinPlot(data,
    x = "x", y = "y", group_by = "group1",
    comparisons = TRUE
)
ViolinPlot(data,
    x = "x", y = "y", sig_label = "p.format",
    facet_by = "group2", comparisons = list(c("A", "B"))
)
ViolinPlot(data,
    x = "x", y = "y", fill_mode = "mean",
    facet_by = "group2", palette = "Blues"
)
ViolinPlot(data,
    x = "x", y = "y", fill_mode = "mean",
    split_by = "group1", palette = c(g1 = "Blues", g2 = "Reds")
)
ViolinPlot(data,
    x = "x", y = "y", stack = TRUE,
    facet_by = "group2", add_box = TRUE, add_bg = TRUE,
    bg_palette = "Paired"
)
```

`ChordPlot` *Chord / Circos plot*

Description

`ChordPlot` is used to create a chord plot to visualize the relationships between two categorical variables. `CircosPlot` is an alias of `ChordPlot`.

Usage

```
ChordPlot(
```

```
  data,
  y = NULL,
  from = NULL,
  from_sep = "_",
  to = NULL,
  to_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  flip = FALSE,
  links_color = c("from", "to"),
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.5,
  labels_rot = FALSE,
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)

CircosPlot(
  data,
  y = NULL,
  from = NULL,
  from_sep = "_",
  to = NULL,
  to_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  flip = FALSE,
  links_color = c("from", "to"),
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.5,
  labels_rot = FALSE,
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
```

```
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
y	A character string specifying the column name of the data frame to plot for the y-axis.
from	A character string of the column name to plot for the source. A character/factor column is expected.
from_sep	A character string to concatenate the columns in from, if multiple columns are provided.
to	A character string of the column name to plot for the target. A character/factor column is expected.
to_sep	A character string to concatenate the columns in to, if multiple columns are provided.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
flip	A logical value to flip the source and target.
links_color	A character string to specify the color of the links. Either "from" or "to".
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
labels_rot	A logical value to rotate the labels by 90 degrees.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.

Value

A combined plot or a list of plots

Examples

```
set.seed(8525)
data <- data.frame(
    nodes1 = sample(c("Soure1", "Source2", "Source3"), 10, replace = TRUE),
    nodes2 = sample(letters[1:3], 10, replace = TRUE),
    y = sample(1:5, 10, replace = TRUE)
)

ChordPlot(data, from = "nodes1", to = "nodes2")
ChordPlot(data, from = "nodes1", to = "nodes2",
          links_color = "to", labels_rot = TRUE)
ChordPlot(data, from = "nodes1", to = "nodes2", y = "y")
ChordPlot(data, from = "nodes1", to = "nodes2", split_by = "y")
ChordPlot(data, from = "nodes1", to = "nodes2", split_by = "y",
          palette = c("1" = "Reds", "2" = "Blues", "3" = "Greens", "4" = "Purp"))
ChordPlot(data, from = "nodes1", to = "nodes2", flip = TRUE)
```

ClustreePlot *Clustree Plot*

Description

A plot visualizing Clusterings at Different Resolutions

Usage

```
ClustreePlot(
  data,
  prefix,
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  palette = "Paired",
  palcolor = NULL,
  edge_palette = "Spectral",
  edge_palcolor = NULL,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
```

```
  expand = c(0.1, 0.1),
  theme = "theme_this",
  theme_args = list(),
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)
```

Arguments

data	A data frame.
prefix	A character string of the prefix of the columns to plot. The columns with the prefix will be used to plot the tree.
flip	A logical value to flip the tree.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
edge_palette	A character string of the palette name to color the edges.
edge_palcolor	A character vector of colors to color the edges.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
expand	The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third

	value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
seed	The random seed to use. Default is 8525.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
N = 100
data <- data.frame(
    p.0.4 = sample(LETTERS[1:5], N, replace = TRUE),
    p.0.5 = sample(LETTERS[1:6], N, replace = TRUE),
    p.0.6 = sample(LETTERS[1:7], N, replace = TRUE),
    p.0.7 = sample(LETTERS[1:8], N, replace = TRUE),
    p.0.8 = sample(LETTERS[1:9], N, replace = TRUE),
    p.0.9 = sample(LETTERS[1:10], N, replace = TRUE),
    p.1 = sample(LETTERS[1:30], N, replace = TRUE),
    split = sample(1:2, N, replace = TRUE)
)

ClustreePlot(data, prefix = "p")
ClustreePlot(data, prefix = "p", flip = TRUE)
ClustreePlot(data, prefix = "p", split_by = "split")
ClustreePlot(data, prefix = "p", split_by = "split",
             palette = c("1" = "Set1", "2" = "Paired"))
```

CorPairsPlot *CorPairsPlot*

Description

Generate a grid of scatter correlation plots for all pairs of variables.

Usage

```
CorPairsPlot(
  data,
  columns = NULL,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  diag_type = NULL,
  diag_args = list(),
  layout = c(".\\", "\\.", "/.", "./"),
  cor_method = c("pearson", "spearman", "kendall"),
  cor_palette = "RdBu",
  cor_palcolor = NULL,
  cor_size = 3,
  cor_format = "corr: {round(corr, 2)}",
  cor_fg = "black",
  cor_bg = "white",
  cor_bg_r = 0.1,
  theme = "theme_this",
  theme_args = list(),
  palette = ifelse(is.null(group_by), "Spectral", "Paired"),
  palcolor = NULL,
  title = NULL,
  subtitle = NULL,
  facet_by = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.

columns	The column names of the data to be plotted. If NULL, all columns, except group_by, will be used.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	The name of the group in the legend.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
diag_type	The type of the diagonal plots. Available types: "density", "violin", "histogram", "box", "none".
diag_args	A list of additional arguments to be passed to the diagonal plots.
layout	The layout of the plots. Available layouts: ".\", "\.", "/.", "./". • '\' or '/' means the diagonal plots are on the top-left to bottom-right diagonal. • '.' means where the scatter plots are.
cor_method	The method to calculate the correlation. Available methods: "pearson", "spearman", "kendall". The correlation will be shown in the other triangle of the scatter plots.
cor_palette	The color palette for the correlation tile plots.
cor_palcolor	Custom colors used to create a color palette for the correlation tile plots.
cor_size	The size of the correlation text.
cor_format	The format of the correlation text. Default is "corr: %.2f". It will be formatted using sprintf(cor_format, corr).
cor_fg	The color of the correlation text.
cor_bg	The background color of the correlation text.
cor_bg_r	The radius of the background of the correlation text.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

seed — The random seed to use. Default is 8525.

combine — Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

... — Additional arguments.

Value

A patch_work::wrap_plots object or a list of them if combine is FALSE.

Examples

```
set.seed(8525)
data <- data.frame(x = rnorm(100))
data$y <- rnorm(100, 10, sd = 0.5)
data$z <- -data$x + data$y + rnorm(100, 20, 1)
data$g <- sample(1:4, 100, replace = TRUE)

CorPairsPlot(data, diag_type = "histogram", diag_args = list(bins = 30, palette = "Paired"),
 layout = "/.")

CorPairsPlot(data, group_by = "g", diag_type = "none", layout = "./",
 theme_args = list(axis.title = element_textbox(
     color = "black", box.color = "grey20", size = 16, halign = 0.5, fill = "grey90",
     linetype = 1, width = grid::unit(1, "npc"), padding = ggplot2::margin(5, 5, 5, 5))))

CorPairsPlot(data, group_by = "g", diag_type = "violin", layout = "\\.",
  cor_format = "{x}\n{y}\ncorr: {round(corr, 2)}")

CorPairsPlot(data, split_by = "g", diag_type = "none", layout = ".\\",
 legend.position = "bottom", legend.direction = "horizontal", group_name = "group")

CorPairsPlot(data, split_by = "g",
 palcolor = list("1" = "red", "2" = "blue", "3" = "green", "4" = "yellow"))
```

CorPlot *CorPlot*

Description

Generate scatter correlation plot for two variables.

Usage

```
CorPlot(
  data,
  x,
  y,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  pt_size = 2,
  pt_shape = 16,
  raster = FALSE,
  alpha = 1,
  raster_dpi = c(512, 512),
  highlight = NULL,
  highlight_color = "black",
  highlight_size = 1,
  highlight_alpha = 1,
  highlight_stroke = 0.8,
  anno_items = c("eq", "r2", "p"),
  anno_size = 3,
  anno_fg = "black",
  anno_bg = "white",
  anno_bg_r = 0.1,
 anno_position = c("topleft", "topright", "bottomleft", "bottomright", "tl", "tr", "bl",
    "br"),
  add_smooth = TRUE,
  smooth_color = "red2",
  smooth_width = 1.5,
  smooth_se = FALSE,
  theme = "theme_this",
  theme_args = list(),
  palette = ifelse(is.null(group_by), "Spectral", "Paired"),
  palcolor = NULL,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
```

```
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	The name of the group in the legend.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
pt_size	The size of the points.
pt_shape	The shape of the points.
raster	Whether to use raster graphics for plotting.
alpha	A numeric value specifying the transparency of the plot.
raster_dpi	The DPI of the raster graphics.
highlight	The items to be highlighted. Could be either a vector of rownames if data has rownames, or a vector of indices, or An expression that can be evaluated by dplyr::filter to get the highlighted items.
highlight_color	The color of the highlighted points.
highlight_size	The size of the highlighted points.
highlight_alpha	The alpha of the highlighted points.

highlight_stroke
: The stroke of the highlighted points.

anno_items
: The items to be annotated on the plot. Available items: "eq", "r2", "p", "spearman", "pearson", "kendall", "n".

anno_size
: The size of the annotation text.

anno_fg
: The color of the annotation text.

anno_bg
: The background color of the annotation text.

anno_bg_r
: The radius of the background of the annotation text.

anno_position
: The position of the annotation text. Available positions: "topleft", "topright", "bottomleft", "bottomright". Shortcuts: "tl", "tr", "bl", "br".

add_smooth
: Whether to add a linear regression line.

smooth_color
: The color of the regression line.

smooth_width
: The width of the regression line.

smooth_se
: Whether to add the standard error band to the regression line.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

seed
: The random seed to use. Default is 8525.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

...
: Additional arguments.

Value

A ggplot object or a list of ggplot objects if `combine` is `FALSE`.

Examples

```
data(iris)
CorPlot(iris, "Sepal.Length", "Sepal.Width", group_by = "Species")
CorPlot(iris, "Sepal.Length", "Sepal.Width", group_by = "Species",
 highlight = 'Species == "setosa"', highlight_stroke = 1.5,
 anno_items = c("eq", "pearson"), anno_position = "bottomright")
CorPlot(iris, "Sepal.Length", "Sepal.Width", facet_by = "Species", facet_scales = "free")
CorPlot(iris, "Sepal.Length", "Sepal.Width", split_by = "Species",
        palette = c(setosa = "Set1", versicolor = "Dark2", virginica = "Paired"))
```

DensityPlot *Density Plot / Histogram*

Description

Density plot and histogram to illustrate the distribution of the data.

Usage

```
DensityPlot(
  data,
  x,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  xtrans = "identity",
  ytrans = "identity",
  split_by = NULL,
  split_by_sep = "_",
```

```
  flip = FALSE,
  position = "identity",
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.5,
  theme = "theme_this",
  theme_args = list(),
  add_bars = FALSE,
  bar_height = 0.025,
  bar_alpha = 1,
  bar_width = 0.1,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  expand = c(bottom = 0, left = 0, right = 0),
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  legend.position = ifelse(is.null(group_by), "none", "right"),
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)

Histogram(
  data,
  x,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  xtrans = "identity",
  ytrans = "identity",
  split_by = NULL,
  split_by_sep = "_",
  flip = FALSE,
  bins = NULL,
  binwidth = NULL,
  trend_skip_zero = FALSE,
  add_bars = FALSE,
  bar_height = 0.025,
  bar_alpha = 1,
```

```
  bar_width = 0.1,
  position = "identity",
  use_trend = FALSE,
  add_trend = FALSE,
  trend_alpha = 1,
  trend_linewidth = 0.8,
  trend_pt_size = 1.5,
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.5,
  theme = "theme_this",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  expand = c(bottom = 0, left = 0, right = 0),
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  legend.position = ifelse(is.null(group_by), "none", "right"),
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to name the legend of group_by
xtrans	A character string specifying the transformation of the x-axis. Default is "identity". Other options see transform of scale_x_continuous.
ytrans	A character string specifying the transformation of the y-axis. Default is "identity". Other options see transform of scale_y_continuous.
split_by	The column(s) to split data by and plot separately.

split_by_sep	The separator for multiple split_by columns. See split_by
flip	A logical value. If TRUE, the plot will be flipped.
position	How should we position the values in each bin? Default is "identity". Unlike the default position = "stack" in ggplot2::geom_histogram or ggplot2::geom_density, the default position is "identity" to show the actual count or density for each group.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
add_bars	A logical value. If TRUE, add lines to the plot to show the data distribution on the bottom.
bar_height	A numeric value specifying the height of the bars. The actual height will be calculated based on the maximum density or count.
bar_alpha	A numeric value specifying the alpha of the bars.
bar_width	A numeric value specifying the width of the bars.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
expand	The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
legend.position	A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.
bins	A numeric value specifying the number of bins for the histogram.
binwidth	A numeric value specifying the width of the bins for the histogram.
trend_skip_zero	A logical value. If TRUE, skip the zero count when drawing the trend line.
use_trend	A logical value. If TRUE, use trend line instead of histogram.
add_trend	A logical value. If TRUE, add trend line to the histogram.
trend_alpha	A numeric value specifying the alpha of the trend line and points
trend_linewidth	A numeric value specifying the width of the trend line
trend_pt_size	A numeric value specifying the size of the trend points

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
data <- data.frame(
    x = c(rnorm(500, -1), rnorm(500, 1)),
    group = rep(c("A", "B"), each = 500),
    facet = sample(c("F1", "F2"), 1000, replace = TRUE)
)

DensityPlot(data, x = "x")
DensityPlot(data, x = "x", group_by = "group", facet_by = "facet")
DensityPlot(data, x = "x", split_by = "facet", add_bars = TRUE)
DensityPlot(data, x = "x", split_by = "facet", add_bars = TRUE,
    palette = c(F1 = "Set1", F2 = "Set2"))
```

```
set.seed(8525)
data <- data.frame(
    x = sample(setdiff(1:100, c(30:36, 50:55, 70:77)), 1000, replace = TRUE),
    group = factor(rep(c("A", "B"), each = 500), levels = c("A", "B")),
    facet = sample(c("F1", "F2"), 1000, replace = TRUE)
)

Histogram(data, x = "x")
Histogram(data, x = "x", group_by = "group")
Histogram(data, x = "x", split_by = "facet", add_bars = TRUE)
Histogram(data, x = "x", group_by = "group", add_trend = TRUE)
Histogram(data, x = "x", group_by = "group", add_trend = TRUE, trend_skip_zero = TRUE)
Histogram(data, x = "x", group_by = "group", split_by = "facet",
 use_trend = TRUE, trend_pt_size = 3)
Histogram(data, x = "x", group_by = "group", split_by = "facet",
 palette = c(F1 = "Paired", F2 = "Spectral"))
```

DimPlot *DimPLot / FeatureDimPlot*

Description

Visualizing the dimension reduction data. `FeatureDimPlot` is used to plot the feature numeric values on the dimension reduction plot.

Usage

```
DimPlot(
  data,
  dims = 1:2,
  group_by,
  group_by_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  pt_size = NULL,
  pt_alpha = 1,
  bg_color = "grey80",
  label_insitu = FALSE,
  show_stat = !identical(theme, "theme_blank"),
  label = FALSE,
  label_size = 4,
  label_fg = "white",
  label_bg = "black",
  label_bg_r = 0.1,
  label_repel = FALSE,
  label_repulsion = 20,
  label_pt_size = 1,
  label_pt_color = "black",
```

```
  label_segment_color = "black",
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  add_mark = FALSE,
  mark_type = c("hull", "ellipse", "rect", "circle"),
  mark_expand = unit(3, "mm"),
  mark_alpha = 0.1,
  mark_linetype = 1,
  stat_by = NULL,
  stat_plot_type = c("pie", "ring", "bar", "line"),
  stat_plot_size = 0.1,
  stat_args = list(palette = "Set1"),
  graph = NULL,
  edge_size = c(0.05, 0.5),
  edge_alpha = 0.1,
  edge_color = "grey40",
  add_density = FALSE,
  density_color = "grey80",
  density_filled = FALSE,
  density_filled_palette = "Greys",
  density_filled_palcolor = NULL,
  lineages = NULL,
  lineages_trim = c(0.01, 0.99),
  lineages_span = 0.75,
  lineages_palette = "Dark2",
  lineages_palcolor = NULL,
  lineages_arrow = arrow(length = unit(0.1, "inches")),
  lineages_linewidth = 1,
  lineages_line_bg = "white",
  lineages_line_bg_stroke = 0.5,
  lineages_whiskers = FALSE,
  lineages_whiskers_linewidth = 0.5,
  lineages_whiskers_alpha = 0.5,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  theme = "theme_this",
  theme_args = list(),
  aspect.ratio = 1,
```

```
  legend.position = "right",
  legend.direction = "vertical",
  raster = NULL,
  raster_dpi = c(512, 512),
  hex = FALSE,
  hex_linewidth = 0.5,
  hex_count = TRUE,
  hex_bins = 50,
  hex_binwidth = NULL,
  palette = "Paired",
  palcolor = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)

FeatureDimPlot(
  data,
  dims = 1:2,
  features,
  split_by = NULL,
  split_by_sep = "_",
  lower_quantile = 0,
  upper_quantile = 0.99,
  lower_cutoff = NULL,
  upper_cutoff = NULL,
  pt_size = NULL,
  pt_alpha = 1,
  bg_color = "grey80",
  bg_cutoff = NULL,
  label_insitu = FALSE,
  show_stat = !identical(theme, "theme_blank"),
  color_name = "",
  label = FALSE,
  label_size = 4,
  label_fg = "white",
  label_bg = "black",
  label_bg_r = 0.1,
  label_repel = FALSE,
  label_repulsion = 20,
  label_pt_size = 1,
  label_pt_color = "black",
  label_segment_color = "black",
  highlight = NULL,
  highlight_alpha = 1,
```

```
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  add_mark = FALSE,
  mark_type = c("hull", "ellipse", "rect", "circle"),
  mark_expand = unit(3, "mm"),
  mark_alpha = 0.1,
  mark_linetype = 1,
  stat_by = NULL,
  stat_plot_type = c("pie", "ring", "bar", "line"),
  stat_plot_size = 0.1,
  stat_args = list(palette = "Set1"),
  graph = NULL,
  edge_size = c(0.05, 0.5),
  edge_alpha = 0.1,
  edge_color = "grey40",
  add_density = FALSE,
  density_color = "grey80",
  density_filled = FALSE,
  density_filled_palette = "Greys",
  density_filled_palcolor = NULL,
  lineages = NULL,
  lineages_trim = c(0.01, 0.99),
  lineages_span = 0.75,
  lineages_palette = "Dark2",
  lineages_palcolor = NULL,
  lineages_arrow = arrow(length = unit(0.1, "inches")),
  lineages_linewidth = 1,
  lineages_line_bg = "white",
  lineages_line_bg_stroke = 0.5,
  lineages_whiskers = FALSE,
  lineages_whiskers_linewidth = 0.5,
  lineages_whiskers_alpha = 0.5,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  theme = "theme_this",
  theme_args = list(),
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  raster = NULL,
```

```
  raster_dpi = c(512, 512),
  hex = FALSE,
  hex_linewidth = 0.5,
  hex_count = FALSE,
  hex_bins = 50,
  hex_binwidth = NULL,
  palette = "Spectral",
  palcolor = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
dims	A character vector of the column names to plot on the x and y axes or a numeric vector of the column indices.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
split_by	A character vector of column names to split the data and plot separately If TRUE, we will split the data by the features. Each feature will be plotted separately.
split_by_sep	The separator for multiple split_by columns. See split_by
pt_size	A numeric value of the point size. If NULL, the point size will be calculated based on the number of data points.
pt_alpha	A numeric value of the point transparency. Default is 1.
bg_color	A character string of the background or NA points. Default is "grey80".
label_insitu	Whether to place the raw labels (group names) in the center of the points with the corresponding group. Default is FALSE, which using numbers instead of raw labels.
show_stat	Whether to show the number of points in the subtitle. Default is TRUE.
label	Whether to show the labels of groups. Default is FALSE.
label_size	A numeric value of the label size. Default is 4.
label_fg	A character string of the label foreground color. Default is "white".
label_bg	A character string of the label background color. Default is "black".
label_bg_r	A numeric value of the background ratio of the labels. Default is 0.1.
label_repel	Whether to repel the labels. Default is FALSE.

label_repulsion
: A numeric value of the label repulsion. Default is 20.

label_pt_size
: A numeric value of the label point size. Default is 1.

label_pt_color
: A character string of the label point color. Default is "black".

label_segment_color
: A character string of the label segment color. Default is "black".

highlight
: A character vector of the row names to highlight. Default is NULL.

highlight_alpha
: A numeric value of the highlight transparency. Default is 1.

highlight_size
: A numeric value of the highlight size. Default is 1.

highlight_color
: A character string of the highlight color. Default is "black".

highlight_stroke
: A numeric value of the highlight stroke. Default is 0.5.

add_mark
: Whether to add mark to the plot. Default is FALSE.

mark_type
: A character string of the mark type. Default is "hull".

mark_expand
: A unit value of the mark expand. Default is 3mm.

mark_alpha
: A numeric value of the mark transparency. Default is 0.1.

mark_linetype
: A numeric value of the mark line type. Default is 1.

stat_by
: A character string of the column name to calculate the statistics. Default is NULL.

stat_plot_type
: A character string of the statistic plot type. Default is "pie".

stat_plot_size
: A numeric value of the statistic plot size. Default is 0.1.

stat_args
: A list of additional arguments to the statistic plot. Default is list(palette = "Set1").

graph
: A character string of column names or the indexes in the data for the graph data. Default is NULL. If "@graph" is provided, the graph data will be extracted from the data attribute 'graph'.

edge_size
: A numeric vector of the edge size range. Default is c(0.05, 0.5).

edge_alpha
: A numeric value of the edge transparency. Default is 0.1.

edge_color
: A character string of the edge color. Default is "grey40".

add_density
: Whether to add density plot. Default is FALSE.

density_color
: A character string of the density color. Default is "grey80".

density_filled
: Whether to fill the density plot. Default is FALSE.

density_filled_palette
: A character string of the filled density palette. Default is "Greys".

density_filled_palcolor
: A character vector of the filled density palette colors. Default is NULL.

lineages
: A character vector of the column names for lineages. Default is NULL.

lineages_trim
: A numeric vector of the trim range for lineages. Default is c(0.01, 0.99).

lineages_span A numeric value of the lineages span. Default is 0.75.

lineages_palette
: A character string of the lineages palette. Default is "Dark2".

lineages_palcolor
: A character vector of the lineages palette colors. Default is NULL.

lineages_arrow An arrow object for the lineages. Default is arrow(length = unit(0.1, "inches")).

lineages_linewidth
: A numeric value of the lineages line width. Default is 1.

lineages_line_bg
: A character string of the lineages line background color. Default is "white".

lineages_line_bg_stroke
: A numeric value of the lineages line background stroke. Default is 0.5.

lineages_whiskers
: Whether to add whiskers to the lineages. Default is FALSE.

lineages_whiskers_linewidth
: A numeric value of the lineages whiskers line width. Default is 0.5.

lineages_whiskers_alpha
: A numeric value of the lineages whiskers transparency. Default is 0.5.

facet_by A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

facet_nrow A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_ncol A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow A logical value indicating whether to fill the plots by row. Default is TRUE.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

raster	Whether to raster the plot. Default is NULL.
raster_dpi	A numeric vector of the raster dpi. Default is c(512, 512).
hex	Whether to use hex plot. Default is FALSE.
hex_linewidth	A numeric value of the hex line width. Default is 0.5.
hex_count	Whether to count the hex.
hex_bins	A numeric value of the hex bins. Default is 50.
hex_binwidth	A numeric value of the hex bin width. Default is NULL.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.
features	A character vector of the column names to plot as features.
lower_quantile, upper_quantile, lower_cutoff, upper_cutoff	Vector of minimum and maximum cutoff values or quantile values for each feature.
bg_cutoff	A numeric value to be used a cutoff to set the feature values to NA. Default is NULL.
color_name	A character string of the color legend name. Default is "".

Value

A ggplot object or wrap_plots object or a list of ggplot objects

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
# generate a PCA dimension data for DimPlot
set.seed(8525)
df <- matrix(c(rnorm(333), rnorm(334, 0.2), rnorm(333, .4)), ncol = 10)
# run PCA
pca <- prcomp(df)
# get coordinates
data <- pca$x[, 1:2]
# kmeans clustering
km <- kmeans(data, 3)
data <- as.data.frame(data)
```

```
data$cluster <- factor(paste0("C", km$cluster))
data$group <- sample(c("A", "B"), nrow(data), replace = TRUE)

graph <- rnorm(nrow(data) * nrow(data))
graph[sample(1:(nrow(data) * nrow(data)), 5000)] <- NA
graph <- matrix(graph, nrow = nrow(data))
rownames(graph) <- colnames(graph) <- rownames(data)

attr(data, "graph") <- graph

data$L1 <- rnorm(nrow(data), 0, 0.1)
data$L2 <- rnorm(nrow(data), 1, 0.2)
data$L3 <- rnorm(nrow(data), 2, 0.3)

DimPlot(data, group_by = "cluster")
DimPlot(data, group_by = "cluster", theme = "theme_blank")
DimPlot(data, group_by = "cluster", theme = ggplot2::theme_classic,
        theme_args = list(base_size = 16), palette = "seurat")
DimPlot(data, group_by = "cluster", raster = TRUE, raster_dpi = 30)
DimPlot(data, group_by = "cluster", highlight = 1:20,
        highlight_color = "red2", highlight_stroke = 0.8)
DimPlot(data, group_by = "cluster", highlight = TRUE, facet_by = "group",
        theme = "theme_blank")
DimPlot(data, group_by = "cluster", label = TRUE)
DimPlot(data, group_by = "cluster", label = TRUE, label_fg = "red",
        label_bg = "yellow", label_size = 5)
DimPlot(data, group_by = "cluster", label = TRUE, label_insitu = TRUE)
DimPlot(data, group_by = "cluster", add_mark = TRUE)
DimPlot(data, group_by = "cluster", add_mark = TRUE, mark_linetype = 2)
DimPlot(data, group_by = "cluster", add_mark = TRUE, mark_type = "ellipse")
DimPlot(data, group_by = "cluster", add_density = TRUE)
DimPlot(data, group_by = "cluster", add_density = TRUE, density_filled = TRUE)
DimPlot(data, group_by = "cluster", add_density = TRUE, density_filled = TRUE,
        density_filled_palette = "Blues", highlight = TRUE)
DimPlot(data, group_by = "cluster", stat_by = "group")
DimPlot(data, group_by = "cluster", stat_by = "group", stat_plot_type = "bar")
DimPlot(data, group_by = "cluster", hex = TRUE)
DimPlot(data, group_by = "cluster", hex = TRUE, hex_bins = 20)
DimPlot(data, group_by = "cluster", hex = TRUE, hex_count = FALSE)
DimPlot(data, group_by = "cluster", graph = "@graph", edge_color = "grey80")
DimPlot(data, group_by = "cluster", lineages = c("L1", "L2", "L3"))
DimPlot(data, group_by = "cluster", lineages = c("L1", "L2", "L3"),
        lineages_whiskers = TRUE)
DimPlot(data, group_by = "cluster", lineages = c("L1", "L2", "L3"),
        lineages_span = 1)
DimPlot(data, group_by = "cluster",  split_by = "cluster",
        palcolor = list(C1 = "red", C2 = "blue", C3 = "green"))


# Feature Dim Plot
FeatureDimPlot(data, features = "L1", pt_size = 2)
FeatureDimPlot(data, features = "L1", pt_size = 2, bg_cutoff = -Inf)
FeatureDimPlot(data, features = "L1", raster = TRUE, raster_dpi = 30)
```

```
FeatureDimPlot(data, features = c("L1", "L2"), pt_size = 2)
FeatureDimPlot(data, features = c("L1"), pt_size = 2, facet_by = "group")
# Can't facet multiple features
FeatureDimPlot(data, features = c("L1", "L2", "L3"), pt_size = 2)
# We can use split_by
FeatureDimPlot(data, features = c("L1", "L2", "L3"), split_by = "group", nrow = 2)
FeatureDimPlot(data, features = c("L1", "L2", "L3"), highlight = TRUE)
FeatureDimPlot(data, features = c("L1", "L2", "L3"), hex = TRUE, hex_bins = 15)
FeatureDimPlot(data, features = c("L1", "L2", "L3"), hex = TRUE, hex_bins = 15,
  split_by = "cluster", palette = list(C1 = "Reds", C2 = "Blues", C3 = "Greens"))
```

DotPlot *Dot Plot / Scatter Plot / Lollipop Plot*

Description

For DotPlot, X-axis and Y-axis could be either numeric or factor/character. When x-axis and y-axis are both numeric, the plot works as a scatter plot. LollipopPlot is an alias of DotPlot when lollipop = TRUE.

Usage

```
DotPlot(
  data,
  x,
  y,
  x_sep = "_",
  y_sep = "_",
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  size_name = NULL,
  fill_name = NULL,
  fill_cutoff_name = NULL,
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  bg_direction = c("vertical", "horizontal", "v", "h"),
  size_by = NULL,
  fill_by = NULL,
  fill_cutoff = NULL,
  fill_reverse = FALSE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
```

```
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  seed = 8525,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)

LollipopPlot(
  data,
  x,
  y,
  y_sep = NULL,
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  size_name = NULL,
  fill_name = NULL,
  fill_cutoff_name = NULL,
  size_by = NULL,
  fill_by = NULL,
  fill_cutoff = NULL,
  fill_reverse = FALSE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
```

```
  x_text_angle = 0,
  seed = 8525,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
x	A character vector specifying the column to use for the x-axis. A numeric column is expected.
y	A character vector specifying the column to use for the y-axis. A factor/character column is expected.
x_sep	A character vector to concatenate multiple columns in x. Default is "_".
y_sep	A character vector to concatenate multiple columns in y. Default is "_".
flip	A logical value indicating whether to flip the x and y axes. Default is FALSE.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
size_name	A character vector specifying the name for the size legend.
fill_name	A character vector specifying the name for the fill legend.
fill_cutoff_name	A character vector specifying the name for the fill cutoff legend.
add_bg	A logical value indicating whether to add a background color to the plot. Default is FALSE.
bg_palette	A character vector specifying the palette for the background color. Default is "stripe".
bg_palcolor	A character vector specifying the color for the background color.
bg_alpha	A numeric value specifying the alpha for the background color. Default is 0.2.
bg_direction	A character vector specifying the direction for the background color. Default is "vertical". Other options are "horizontal". "h" and "v" are also accepted.
size_by	Which column to use as the size of the dots. It must be a numeric column. If not provided, the size will be the count of the instances for each 'y' in 'x'. For 'ScatterPlot', it can be a single numeric value to specify the size of the dots.

fill_by	Which column to use as the fill the dots. It must be a numeric column. If not provided, all dots will be filled with the same color at the middle of the palette.
fill_cutoff	A numeric value specifying the cutoff for the fill column.
fill_reverse	A logical value indicating whether to reverse the fill direction. Default is FALSE. By default, the fill direction is "up". If TRUE, the fill direction is "down". When the direction is "up", the values less than the cutoff will be filled with grey. When the direction is "down", the values greater than the cutoff will be filled with grey.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
x_text_angle	A numeric value specifying the angle of the x-axis text.
seed	The random seed to use. Default is 8525.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.

keep_empty	A logical value indicating whether to keep empty groups. If FALSE, empty groups will be removed.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
mtcars <- datasets::mtcars
mtcars$carb <- factor(mtcars$carb)
mtcars$gear <- factor(mtcars$gear)
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18)
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, add_bg = TRUE)
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, add_bg = TRUE,
        bg_direction = "h")
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, facet_by = "cyl")
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, facet_by = "cyl",
        facet_scales = "free_x")
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, split_by = "cyl")
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, split_by = "cyl",
        palette = list("4" = "Set1", "6" = "Paired", "8" = "Reds"))
# works as a scatter plot
DotPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, fill_cutoff_name = "Small mpgs")
LollipopPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
             fill_by = "mpg")
LollipopPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
             fill_by = "mpg", fill_cutoff = 18, facet_by = "cyl",
             facet_scales = "free_y")
LollipopPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
             split_by = "vs", palette = list("0" = "Reds", "1" = "Blues"))
```

element_textbox *Theme element that add a box to the text*

Description

Code grabbed from the ggtext package. See the original code at: https://github.com/wilkelab/ggtext
This is used to create a text box around the text, primarily to be used in CorPairsPlot.

Usage

```
element_textbox(
  family = NULL,
  face = NULL,
  size = NULL,
  colour = NULL,
  fill = NULL,
  box.colour = NULL,
  linetype = NULL,
  linewidth = NULL,
  hjust = NULL,
  vjust = NULL,
  halign = NULL,
  valign = NULL,
  lineheight = NULL,
  margin = NULL,
  padding = NULL,
  width = NULL,
  height = NULL,
  minwidth = NULL,
  maxwidth = NULL,
  minheight = NULL,
  maxheight = NULL,
  r = NULL,
  orientation = NULL,
  color = NULL,
  box.color = NULL,
  debug = FALSE,
  inherit.blank = FALSE
)

## S3 method for class 'element_textbox'
element_grob(
  element,
  label = "",
  x = NULL,
  y = NULL,
  family = NULL,
```

```
  face = NULL,
  colour = NULL,
  size = NULL,
  hjust = NULL,
  vjust = NULL,
  lineheight = NULL,
  margin = NULL,
  ...
)
```

Arguments

family	Font family
face	Font face
size	Font size (in pt)
colour, color	Text color
fill	Fill color of the enclosing box
box.colour, box.color	Line color of the enclosing box (if different from the text color)
linetype	Line type of the enclosing box (like lty in base R)
linewidth	Line width of the enclosing box (measured in mm, just like size in ggplot2::element_line()).
hjust	Horizontal justification
vjust	Vertical justification
halign	Horizontal justification
valign	Vertical justification
lineheight	Line height, in multiples of the font size
padding, margin	Padding and margins around the text box. See gridtext::textbox_grob() for details.
width, height	Unit objects specifying the width and height of the textbox, as in gridtext::textbox_grob().
minwidth, minheight, maxwidth, maxheight	Min and max values for width and height. Set to NULL to impose neither a minimum nor a maximum.
r	Unit value specifying the corner radius of the box
orientation	Orientation of the text box. See gridtext::textbox_grob() for details.
debug	Not implemented.
inherit.blank	See ggplot2::margin() for details.
element	A theme element created by element_textbox().
label	Text to display in the textbox.
x, y	Position of the textbox.
...	Other arguments passed to gridtext::textbox_grob().

Value

A ggplot2 theme element that can be used inside a ggplot2::theme() call.

enrich_example *A example of clusterProfiler enrichment result*

Description

A example of clusterProfiler enrichment result

Examples

```
## Not run:
if (interactive()) {
  data(geneList, package="DOSE")
  de <- names(geneList)[abs(geneList) > 1.5]
 enrich_example <- clusterProfiler::enrichPathway(gene=de, pvalueCutoff = 0.05, readable=TRUE)
  enrich_example <- as.data.frame(enrich_example)
}

## End(Not run)
```

enrich_multidb_example *A example of clusterProfiler enrichment result with multiple databases*

Description

A example of clusterProfiler enrichment result with multiple databases

Examples

```
## Not run:
if (interactive()) {
  data(enrich_example, package="plotthis")
  enrich_example$Database <- "DB1"
  enrich_example2 <- enrich_example
  enrich_example2$Database <- "DB2"
  enrich_example2$ID <- paste0(enrich_example2$ID, "_DB2")
  enrich_example2$Description <- paste0(enrich_example2$Description, " (DB2)")
  enrich_multidb_example <- rbind(enrich_example, enrich_example2)
}

## End(Not run)
```

gsea_example *A example of GSEA result from fgsea package*

Description

A example of GSEA result from fgsea package

Examples

```
## Not run:
if (interactive()) {
 set.seed(1234)
 data(geneList, package="DOSE")
 gsea_example <- DOSE::gseDO(geneList)
 gene_ranks <- gsea_example@geneList
 gene_sets <- gsea_example@geneSets
 gsea_example_pos <- gsea_example[gsea_example$p.adjust < 0.05 & gsea_example$NES > 0, ]
 gsea_example_neg <- gsea_example[gsea_example$p.adjust < 0.05 & gsea_example$NES < 0, ]
 gsea_example <- rbind(
     gsea_example_pos[sample(1:nrow(gsea_example_pos), 5), ],
     gsea_example_pos[sample(1:nrow(gsea_example_pos), 5), ]
 )

 attr(gsea_example, "gene_ranks") <- gene_ranks
 attr(gsea_example, "gene_sets") <- gene_sets[gsea_example$ID]
}

## End(Not run)
```

Heatmap *Heatmap*

Description

Heatmap is a popular way to visualize data in matrix format. It is widely used in biology to visualize gene expression data in microarray and RNA-seq data. The heatmap is a matrix where rows represent the samples and columns represent the features. The color of each cell represents the value of the feature in the sample. The color can be continuous or discrete. The heatmap can be split by the columns or rows to show the subgroups in the data. The heatmap can also be annotated by the columns or rows to show the additional information of the samples or features.

Usage

```
Heatmap(
  data,
  rows,
  columns_by,
```

```
 rows_name = "rows",
 columns_name = "columns",
 split_by = NULL,
 split_by_sep = "_",
 split_rows_data = FALSE,
 name = "value",
 border = TRUE,
 rows_palette = "Paired",
 rows_palcolor = NULL,
 title = NULL,
 pie_group_by = NULL,
 pie_group_by_sep = "_",
 pie_palette = "Spectral",
 pie_palcolor = NULL,
 pie_size = NULL,
 pie_name = NULL,
 pie_size_name = "size",
 pie_values = "count",
 legend_items = NULL,
 legend_discrete = FALSE,
 lower_quantile = 0,
 upper_quantile = 0.99,
 lower_cutoff = NULL,
 upper_cutoff = NULL,
 columns_by_sep = "_",
 columns_split_by = NULL,
 columns_split_name = NULL,
 columns_palette = "Paired",
 columns_palcolor = NULL,
 columns_split_by_sep = "_",
 columns_split_palette = "simspec",
 columns_split_palcolor = NULL,
 rows_data = NULL,
 rows_split_by = NULL,
 rows_split_by_sep = "_",
 rows_split_palette = "simspec",
 rows_split_palcolor = NULL,
 column_name_annotation = TRUE,
column_name_legend = isFALSE(show_column_names) && !identical(legend.position, "none"),
 row_name_annotation = TRUE,
row_name_legend = isFALSE(show_row_names) && !identical(legend.position, "none"),
 cluster_columns = TRUE,
 cluster_rows = TRUE,
 show_row_names = !row_name_annotation,
 show_column_names = !column_name_annotation,
 column_title = character(0),
 row_title = character(0),
 na_col = "grey85",
```

```
  row_names_side = "right",
  column_names_side = "bottom",
  bars_sample = 100,
  flip = FALSE,
  label_size = 10,
  label_cutoff = NULL,
  label_accuracy = 0.01,
  layer_fun_callback = NULL,
  cell_type = c("tile", "bars", "label", "dot", "violin", "boxplot", "pie"),
  cell_agg = mean,
  add_bg = FALSE,
  bg_alpha = 0.5,
  violin_fill = NULL,
  boxplot_fill = NULL,
  dot_size = 8,
  dot_size_name = "size",
  column_annotation = NULL,
  column_annotation_side = "top",
  column_annotation_palette = "Paired",
  column_annotation_palcolor = NULL,
  column_annotation_type = "auto",
  column_annotation_params = list(),
  column_annotation_agg = NULL,
  row_annotation = NULL,
  row_annotation_side = "left",
  row_annotation_palette = "Paired",
  row_annotation_palcolor = NULL,
  row_annotation_type = "auto",
  row_annotation_params = list(),
  row_annotation_agg = NULL,
  add_reticle = FALSE,
  reticle_color = "grey",
  palette = "RdBu",
  palcolor = NULL,
  alpha = 1,
  legend.position = "right",
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame used to create the heatmap. The data should be in a long form where each row represents a instance in the heatmap. The rows should be mul-

tiple columns if you want to plot as rows, which you can refer as "features".

`rows` A character string/vector of the column name(s) to plot for the rows Multiple columns in the data frame can be used as the rows.

`columns_by` A character string of the column name to plot for the columns A character/factor column is expected. Multiple columns are also supported. If so, `NA`s can be used to exclude rows in that group. If there is a single value in each group (other than `NA`), the group name, rather than the value, will be shown in the column group annotation.

`rows_name` A character string specifying the name of rows, which will be shown in the row group annotation and the legend of it.

`columns_name` A character string specifying the name of columns, which will be shown in the column group annotation and the legend of it. Only used when `columns_by` has multiple columns.

`split_by` The column(s) to split data by and plot separately.

`split_by_sep` The separator for multiple split_by columns. See `split_by`

`split_rows_data`
A logical value indicating whether to split the rows data as well using 'split_by' and 'split_by_sep'.

`name` A character string specifying the name of the main legend of the heatmap

`border` A logical value indicating whether to draw the border of the heatmap. If TRUE, the borders of the slices will be also drawn.

`rows_palette` A character string specifying the palette of the row group annotation. The default is "Paired".

`rows_palcolor` A character vector of colors to override the palette of the row group annotation.

`title` A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

`pie_group_by` A character string of the column name to group the data for the pie chart.

`pie_group_by_sep`
A character string to concatenate the columns in `pie_group_by` if there are multiple columns.

`pie_palette` A character string specifying the palette of the pie chart.

`pie_palcolor` A character vector of colors to override the palette of the pie chart.

`pie_size` A numeric value or a function specifying the size of the pie chart. If it is a function, the function should take `count` as the argument and return the size.

`pie_name` A character string specifying the name of the legend for the pie chart.

`pie_size_name` A character string specifying the name of the legend for the pie size.

`pie_values` How to calculate the values for the pie chart. Default is "count". Other options are "value" and a function.

- "count": Count the number of instances in `pie_group_by`.
- "value": Use the values in the columns specified by `rows` as the values. If there are multiple values for a single group in `pie_group_by`, a function should be provided to aggregate the values. If not, the sum of the values will be used and a warning will be shown.

- A function: The function should take a vector of values as the argument and return a single value, for each group in `pie_group_by`.

`legend_items` A numeric vector with names to specifiy the items in the main legend. The names will be working as the labels of the legend items.

`legend_discrete`
A logical value indicating whether the main legend is discrete.

`lower_quantile`, `upper_quantile`, `lower_cutoff`, `upper_cutoff`
Vector of minimum and maximum cutoff values or quantile values for each feature. It's applied to aggregated values when aggregated values are used (e.g. plot_type tile, label, etc). It's applied to raw values when raw values are used (e.g. plot_type bars, etc).

`columns_by_sep` A character string to concatenate the columns in `columns_by` if there are multiple columns.

`columns_split_by`
A character string of the column name to split the heatmap columns into slices. A character/factor column or multiple columns are expected.

`columns_split_name`
A character string specifying the name of the column split annotation.

`columns_palette`
A character string specifying the palette of the column group annotation. The default is "Paired".

`columns_palcolor`
A character vector of colors to override the palette of the column group annotation.

`columns_split_by_sep`
A character string to concatenate the columns in `columns_split_by` if there are multiple columns.

`columns_split_palette`
A character string specifying the palette of the column split annotation. The default is "simspec".

`columns_split_palcolor`
A character vector of colors to override the palette of the column split annotation.

`rows_data` A character string of the column name to use as the data for the row group annotation. If it starts with "@", it will be treated as an attribute of the data.

`rows_split_by` A character string of the column name to split the heatmap rows into slices. A character/factor column or multiple columns are expected.

`rows_split_by_sep`
A character string to concatenate the columns in `rows_split_by` if there are multiple columns.

`rows_split_palette`
A character string specifying the palette of the row split annotation. The default is "simspec".

`rows_split_palcolor`
A character vector of colors to override the palette of the row split annotation.

column_name_annotation
: A logical value indicating whether to add the column annotation for the column names. which is a simple annotaion indicating the column names.

column_name_legend
: A logical value indicating whether to show the legend of the column name annotation.

row_name_annotation
: A logical value indicating whether to add the row annotation for the row names. which is a simple annotaion indicating the row names.

row_name_legend
: A logical value indicating whether to show the legend of the row name annotation.

cluster_columns
: A logical value indicating whether to cluster the columns. If TRUE and columns_split_by is provided, the clustering will only be applied to the columns within the same split.

cluster_rows
: A logical value indicating whether to cluster the rows. If TRUE and rows_split_by is provided, the clustering will only be applied to the rows within the same split.

show_row_names
: A logical value indicating whether to show the row names. If TRUE, the legend of the row group annotation will be hidden.

show_column_names
: A logical value indicating whether to show the column names. If TRUE, the legend of the column group annotation will be hidden.

column_title
: A character string/vector of the column name(s) to use as the title of the column group annotation.

row_title
: A character string/vector of the column name(s) to use as the title of the row group annotation.

na_col
: A character string specifying the color for missing values. The default is "grey85".

row_names_side
: A character string specifying the side of the row names. The default is "right".

column_names_side
: A character string specifying the side of the column names. The default is "bottom".

bars_sample
: An integer specifying the number of samples to draw the bars.

flip
: A logical value indicating whether to flip the heatmap.

label_size
: A numeric value specifying the size of the labels when `cell_type = "label"`.

label_cutoff
: A numeric value specifying the cutoff to show the labels when `cell_type = "label"`.

label_accuracy
: A numeric value specifying the accuracy of the labels when `cell_type = "label"`.

layer_fun_callback
: A function to add additional layers to the heatmap. The function should have the following arguments: `j`, `i`, `x`, `y`, `w`, `h`, `fill`, `sr` and `sc`. Please also refer to the `layer_fun` argument in `ComplexHeatmap::Heatmap`.

`cell_type` A character string specifying the type of the heatmap cells. The default is values. Other options are "bars", "label", "dot", "violin", "boxplot". Note that for pie chart, the values under columns specified by `rows` will not be used directly. Instead, the values will just be counted in different `pie_group_by` groups. `NA` values will not be counted.

`cell_agg` A function to aggregate the values in the cell, for the cell type "tile" and "label". The default is `mean`.

`add_bg` A logical value indicating whether to add a background to the heatmap. Does not work with `cell_type = "bars"` or `cell_type = "tile"`.

`bg_alpha` A numeric value between 0 and 1 specifying the transparency of the background.

`violin_fill` A character vector of colors to override the fill color of the violin plot. If NULL, the fill color will be the same as the annotion.

`boxplot_fill` A character vector of colors to override the fill color of the boxplot. If NULL, the fill color will be the same as the annotion.

`dot_size` A numeric value specifying the size of the dot or a function to calculate the size from the values in the cell.

`dot_size_name` A character string specifying the name of the legend for the dot size.

`column_annotation`
A character string/vector of the column name(s) to use as the column annotation. Or a list with the keys as the names of the annotation and the values as the column names.

`column_annotation_side`
A character string specifying the side of the column annotation. Could be a list with the keys as the names of the annotation and the values as the sides.

`column_annotation_palette`
A character string specifying the palette of the column annotation. The default is "Paired". Could be a list with the keys as the names of the annotation and the values as the palettes.

`column_annotation_palcolor`
A character vector of colors to override the palette of the column annotation. Could be a list with the keys as the names of the annotation and the values as the palcolors.

`column_annotation_type`
A character string specifying the type of the column annotation. The default is "auto". Other options are "simple", "pie", "ring", "bar", "violin", "boxplot", "density". Could be a list with the keys as the names of the annotation and the values as the types. If the type is "auto", the type will be determined by the type and number of the column data.

`column_annotation_params`
A list of parameters passed to the annotation function. Could be a list with the keys as the names of the annotation and the values as the parameters.

`column_annotation_agg`
A function to aggregate the values in the column annotation.

`row_annotation` A character string/vector of the column name(s) to use as the row annotation. Or a list with the keys as the names of the annotation and the values as the column names.

row_annotation_side
: A character string specifying the side of the row annotation. Could be a list with the keys as the names of the annotation and the values as the sides.

row_annotation_palette
: A character string specifying the palette of the row annotation. The default is "Paired". Could be a list with the keys as the names of the annotation and the values as the palettes.

row_annotation_palcolor
: A character vector of colors to override the palette of the row annotation. Could be a list with the keys as the names of the annotation and the values as the palcolors.

row_annotation_type
: A character string specifying the type of the row annotation. The default is "auto". Other options are "simple", "pie", "ring", "bar", "violin", "boxplot", "density". Could be a list with the keys as the names of the annotation and the values as the types. If the type is "auto", the type will be determined by the type and number of the row data.

row_annotation_params
: A list of parameters passed to the annotation function. Could be a list with the keys as the names of the annotation and the values as the parameters.

row_annotation_agg
: A function to aggregate the values in the row annotation.

add_reticle
: A logical value indicating whether to add a reticle to the heatmap.

reticle_color
: A character string specifying the color of the reticle.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

alpha
: A numeric value specifying the transparency of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

seed
: The random seed to use. Default is 8525.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

...
: Additional arguments.

Value

The heatmap(s). When `split_by` is not provided, the function returns a single heatmap (ComplexHeatmap::Heatmap-class object). When `split_by` is provided, the function returns a combined plot of multiple heatmaps wrapped by patchwork::wrap_plots if `combine = TRUE`, otherwise, it returns a list of heatmaps.

See Also

anno_simple, anno_points, anno_lines, anno_pie, anno_violin, anno_boxplot, anno_density

Examples

```
set.seed(8525)
data <- data.frame(
   F1 = rnorm(100, 0.1),
   F2 = rnorm(100, 0.2),
   F3 = rnorm(100),
   F4 = rnorm(100, 0.3),
   F5 = rnorm(100, -0.1),
   F6 = rnorm(100, -0.2),
   c = sample(letters[1:8], 100, replace = TRUE),
   s = sample(LETTERS[1:2], 100, replace = TRUE),
   p = sample(c("X", "Y", "Z"), 100, replace = TRUE),
   a = sample(1:5, 100, replace = TRUE),
   p1 = c(sample(c(1, NA), 100, replace = TRUE)),
   p2 = c(sample(c(1, NA), 100, replace = TRUE)),
   p3 = c(sample(c(1, NA), 100, replace = TRUE))
)
rows <- c("F1", "F2", "F3", "F4", "F5", "F6")
rows_data <- data.frame(
   rows = rep(c("F1", "F2", "F3", "F4", "F5", "F6"), each = 10),
   rows1 = rep(c("F1", "F2", "F3", "F4", "F5", "F6"), each = 10),
   rs = rep(letters[1:2], each = 30),
   rp = sample(c("X", "Y", "Z"), 60, replace = TRUE),
   rv = rnorm(60, 0.5)
)

Heatmap(data, rows = rows, columns_by = "c")
Heatmap(data, rows = list(RG1 = c("F1", "F2", "F3"), RG2 = c("F4", "F5", "F6")),
   columns_by = "c")
# Multiple columns_by, each as a split
Heatmap(data, rows = rows, columns_by = c("c", "s"), columns_by_sep = "/")
Heatmap(data, rows = rows, columns_by = c("p1", "p2", "p3"))
Heatmap(data, rows = rows, columns_by = "c", split_by = "s")
Heatmap(data, rows = rows, columns_by = "c", columns_split_by = "s")
Heatmap(data, rows = rows, columns_by = "c", columns_split_by = "s",
        rows_data = rows_data, rows_split_by = "rs")
Heatmap(data, rows = rows, columns_by = "c", columns_split_by = "s",
        rows_data = rows_data, rows_split_by = "rs", flip = TRUE)
Heatmap(data, rows = rows, columns_by = "c", cell_type = "bars")
Heatmap(data, rows = rows, columns_by = "c", cell_type = "bars",
        bars_sample = 3)
```

```
Heatmap(data, rows = rows, columns_by = "c", cell_type = "label")
Heatmap(data, rows = rows, columns_by = "c", cell_type = "label",
        label_cutoff = 0)
Heatmap(data, rows = rows, columns_by = "c", cell_type = "dot")
Heatmap(data, rows = rows, columns_by = "c", cell_type = "dot",
        dot_size = mean)
Heatmap(data, rows = rows, columns_by = "c", cell_type = "dot",
        dot_size = mean, row_name_annotation = FALSE, column_name_annotation = FALSE,
        row_names_side = "left", cluster_rows = FALSE, cluster_columns = FALSE)
Heatmap(data, rows = rows, columns_by = "c", cell_type = "dot",
        dot_size = mean, add_bg = TRUE)
Heatmap(data, rows = rows, columns_by = "c", cell_type = "dot",
        dot_size = mean, add_reticle = TRUE)
Heatmap(data, cluster_rows = FALSE, cluster_columns = FALSE, columns_by = "p",
   rows = c("p1", "p2", "p3"), name = "Category", pie_group_by = "c",
   cell_type = "pie")
Heatmap(data, cluster_rows = FALSE, cluster_columns = FALSE, columns_by = "p",
   rows = c("p1", "p2", "p3"), name = "Category", pie_group_by = "c",
   cell_type = "pie", pie_size = sqrt, add_bg = TRUE)
Heatmap(data, rows = rows, columns_by = "c", cell_type = "violin")
Heatmap(data, rows = rows, columns_by = "c", cell_type = "boxplot")
Heatmap(data, rows = rows, columns_by = "c", rows_data = rows_data,
  column_annotation = list(p1 = "p", p2 = "p", F1 = "F1"),
  column_annotation_type = list(p1 = "ring", p2 = "bar", F1 = "violin"),
  column_annotation_params = list(
     p1 = list(height = grid::unit(10, "mm"), show_legend = FALSE),
     F1 = list(height = grid::unit(18, "mm"))),
  rows_split_by = "rs",
  row_annotation = c("rp", "rv", "rows1"),
  row_annotation_side = "right",
  row_annotation_type = list(rp = "pie", rv = "density", rows1 = "simple"),
  row_annotation_params = list(rp = list(width = grid::unit(12, "mm"))),
  show_row_names = TRUE, show_column_names = TRUE)
Heatmap(data, rows = rows, columns_by = "c", rows_data = rows_data,
  column_annotation = list(p1 = "p", p2 = "p", F1 = "F1"),
  column_annotation_type = list(p1 = "ring", p2 = "bar", F1 = "violin"),
  column_annotation_params = list(
     p1 = list(height = grid::unit(10, "mm"), show_legend = FALSE),
     F1 = list(height = grid::unit(18, "mm"))),
  rows_split_by = "rs",
  row_annotation = c("rp", "rv", "rows1"),
  row_annotation_side = "right",
  row_annotation_type = list(rp = "pie", rv = "density", rows1 = "simple"),
  row_annotation_params = list(rp = list(width = grid::unit(12, "mm"))),
  show_row_names = TRUE, show_column_names = TRUE, flip = TRUE)
Heatmap(data, rows = rows, columns_by = "c", split_by = "p",
        palette = list(X = "Reds", Y = "Blues", Z = "Purp"))
```

LinePlot *Line Plot*

Description

Visualizing the change of a numeric value over the progression of a categorical variable.

Usage

```
LinePlot(
  data,
  x,
  y = NULL,
  group_by = NULL,
  group_by_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  fill_point_by_x_if_no_group = TRUE,
  color_line_by_x_if_no_group = TRUE,
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_errorbars = FALSE,
  errorbar_width = 0.1,
  errorbar_alpha = 1,
  errorbar_color = "grey30",
  errorbar_linewidth = 0.75,
  errorbar_min = NULL,
  errorbar_max = NULL,
  errorbar_sd = NULL,
  pt_alpha = 1,
  pt_size = 5,
  line_type = "solid",
  line_width = 1,
  line_alpha = 0.8,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  facet_by = NULL,
  facet_scales = "fixed",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  facet_nrow = NULL,
  facet_ncol = NULL,
```

```
  facet_byrow = TRUE,
  facet_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  seed = 8525,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	A character string specifying the separator to use when concatenating multiple columns.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
fill_point_by_x_if_no_group	A logical value indicating whether to color the points by the x-axis values when there is no group_by column.
color_line_by_x_if_no_group	A logical value indicating whether to color the lines by the x-axis values
add_bg	A logical value indicating whether to add a background to the plot.
bg_palette	The palette to use for the background.
bg_palcolor	The color to use for the background.
bg_alpha	The alpha value of the background.
add_errorbars	A logical value indicating whether to add error bars to the plot.
errorbar_width	The width of the error bars.
errorbar_alpha	The alpha value of the error bars.
errorbar_color	The color to use for the error bars. If "line", the error bars will be colored the same as the lines.
errorbar_linewidth	The line width of the error bars.
errorbar_min	The column in the data frame containing the lower bound of the error bars.
errorbar_max	The column in the data frame containing the upper bound of the error bars.

errorbar_sd
: The column in the data frame containing the standard deviation of the error bars. If errorbar_min and errorbar_max are not provided, this column will be used to calculate the error bars. errorbar_min = y - errorbar_sd, errorbar_max = y + errorbar_sd. If errorbar_min and errorbar_max are provided, this column will be ignored.

pt_alpha
: The alpha value of the points.

pt_size
: The size of the points.

line_type
: The type of line to draw.

line_width
: The width of the line.

line_alpha
: The alpha value of the line.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

x_text_angle
: A numeric value specifying the angle of the x-axis text.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

facet_args
: A list of arguments to pass to `ggplot2::facet_wrap()` or `ggplot2::facet_grid()`. when there is no group_by column.

title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
keep_empty	A logical value indicating whether to keep empty groups. If FALSE, empty groups will be removed.
seed	The random seed to use. Default is 8525.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- data.frame(
   x = factor(c("A", "B", "C", "D", "A", "B", "C", "D"), levels = LETTERS[1:6]),
   y = c(10, 8, 16, 4, 6, 12, 14, 2),
   group = c("G1", "G1", "G1", "G1", "G2", "G2", "G2", "G2"),
   facet = c("F1", "F1", "F2", "F2", "F3", "F3", "F4", "F4")
)

LinePlot(data, x = "x", y = "y")
LinePlot(data, x = "x", y = "y", group_by = "group")
LinePlot(data, x = "x", y = "y", group_by = "group", add_bg = TRUE)
LinePlot(data, x = "x", y = "y", group_by = "group", facet_by = "facet")
LinePlot(data, x = "x", y = "y", group_by = "group", split_by = "facet")
LinePlot(data, x = "x", y = "y", split_by = "group",
         palcolor = list(G1 = c("red", "blue"), G2 = c("green", "black")))
```

Network *Network*

Description

Plot a network graph

Usage

```
Network(
  links,
  nodes = NULL,
  split_by = NULL,
  split_by_sep = "_",
  split_nodes = FALSE,
```

```
  from = NULL,
  from_sep = "_",
  to = NULL,
  to_sep = "_",
  node_by = NULL,
  node_by_sep = "_",
  link_weight_by = 2,
  link_weight_name = NULL,
  link_type_by = "solid",
  link_type_name = NULL,
  node_size_by = 15,
  node_size_name = NULL,
  node_color_by = "black",
  node_color_name = NULL,
  node_shape_by = 21,
  node_shape_name = NULL,
  node_fill_by = "grey20",
  node_fill_name = NULL,
  link_alpha = 1,
  node_alpha = 0.95,
  node_stroke = 1.5,
  cluster_scale = c("fill", "color", "shape"),
  node_size_range = c(5, 20),
  link_weight_range = c(0.5, 5),
  link_arrow_offset = 20,
  link_curvature = 0,
  link_color_by = "from",
  link_color_name = NULL,
  palette = "Paired",
  palcolor = NULL,
  link_palette = ifelse(link_color_by %in% c("from", "to"), palette, "Set1"),
  link_palcolor = if (link_color_by %in% c("from", "to")) palcolor else NULL,
  directed = TRUE,
  layout = "circle",
  cluster = "none",
  add_mark = FALSE,
  mark_expand = ggplot2::unit(10, "mm"),
  mark_type = c("hull", "ellipse", "rect", "circle"),
  mark_alpha = 0.1,
  mark_linetype = 1,
  add_label = TRUE,
  label_size = 3,
  label_fg = "white",
  label_bg = "black",
  label_bg_r = 0.1,
 arrow = ggplot2::arrow(type = "closed", length = ggplot2::unit(0.1, "inches")),
  title = NULL,
  subtitle = NULL,
```

```
  xlab = NULL,
  ylab = NULL,
  aspect.ratio = 1,
  theme = "theme_this",
  theme_args = list(),
  legend.position = "right",
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

links	A data frame containing the links between nodes.
nodes	A data frame containing the nodes. This is optional. The names of the nodes are extracted from the links data frame. If "@nodes" is provided, the nodes data frame will be extracted from the attribute nodes of the links data frame.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
split_nodes	A logical value specifying whether to split the nodes data. If TRUE, the nodes data will also be split by the split_by column.
from	A character string specifying the column name of the links data frame for the source nodes. Default is the first column of the links data frame.
from_sep	A character string to concatenate the columns in from, if multiple columns are provided.
to	A character string specifying the column name of the links data frame for the target nodes. Default is the second column of the links data frame.
to_sep	A character string to concatenate the columns in to, if multiple columns are provided.
node_by	A character string specifying the column name of the nodes data frame for the node names. Default is the first column of the nodes data frame.
node_by_sep	A character string to concatenate the columns in node_by, if multiple columns are provided.
link_weight_by	A numeric value or a character string specifying the column name of the links data frame for the link weight. If a numeric value is provided, all links will have the same weight. This determines the width of the links.
link_weight_name	A character string specifying the name of the link weight in the legend.
link_type_by	A character string specifying the type of the links. This can be "solid", "dashed", "dotted", or a column name from the links data frame. It has higher priority when it is a column name.

link_type_name A character string specifying the name of the link type in the legend.

node_size_by A numeric value or a character string specifying the column name of the nodes data frame for the node size. If a numeric value is provided, all nodes will have the same size.

node_size_name A character string specifying the name of the node size in the legend.

node_color_by A character string specifying the color of the nodes. This can be a color name, a hex code, or a column name from the nodes data frame. It has higher priority when it is a column name.

node_color_name
A character string specifying the name of the node color in the legend.

node_shape_by A numeric value or a character string specifying the column name of the nodes data frame for the node shape. If a numeric value is provided, all nodes will have the same shape.

node_shape_name
A character string specifying the name of the node shape in the legend.

node_fill_by A character string specifying the fill color of the nodes. This can be a color name, a hex code, or a column name from the nodes data frame. It has higher priority when it is a column name.

node_fill_name A character string specifying the name of the node fill in the legend.

link_alpha A numeric value specifying the transparency of the links.

node_alpha A numeric value specifying the transparency of the nodes. It only works when the nodes are filled.

node_stroke A numeric value specifying the stroke of the nodes.

cluster_scale A character string specifying how to scale the clusters. It can be "fill", "color", or "shape".

node_size_range
A numeric vector specifying the range of the node size.

link_weight_range
A numeric vector specifying the range of the link weight.

link_arrow_offset
A numeric value specifying the offset of the link arrows. So that they won't overlap with the nodes.

link_curvature A numeric value specifying the curvature of the links.

link_color_by A character string specifying the colors of the link. It can be:

- "from" means the color of the link is determined by the source node.
- "to" means the color of the link is determined by the target node.
- Otherwise, the color of the link is determined by the column name from the links data frame.

link_color_name
A character string specifying the name of the link color in the legend. Only used when `link_color_by` is a column name.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

link_palette — A character string specifying the palette of the links. When link_color_by is "from" or "to", the palette of the links defaults to the palette of the nodes.

link_palcolor — A character vector specifying the colors of the link palette. When link_color_by is "from" or "to", the colors of the link palette defaults to the colors of the node palette.

directed — A logical value specifying whether the graph is directed.

layout — A character string specifying the layout of the graph. It can be "circle", "tree", "grid", or a layout function from igraph.

cluster — A character string specifying the clustering method. It can be "none", "fast_greedy", "walktrap", "edge_betweenness", "infomap", or a clustering function from igraph.

add_mark — A logical value specifying whether to add mark for the clusters to the plot.

mark_expand — A unit value specifying the expansion of the mark.

mark_type — A character string specifying the type of the mark. It can be "hull", "ellipse", "rect", "circle", or a mark function from ggforce.

mark_alpha — A numeric value specifying the transparency of the mark.

mark_linetype — A numeric value specifying the line type of the mark.

add_label — A logical value specifying whether to add label to the nodes to the plot.

label_size — A numeric value specifying the size of the label.

label_fg — A character string specifying the foreground color of the label.

label_bg — A character string specifying the background color of the label.

label_bg_r — A numeric value specifying the background ratio of the label.

arrow — An arrow object for the links.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
actors <- data.frame(
  name = c("Alice", "Bob", "Cecil", "David", "Esmeralda"),
  age = c(48, 33, 45, 34, 21),
  shape = c(21, 22, 21, 22, 23),
  gender = c("F", "M", "F", "M", "F")
)
relations <- data.frame(
  from = c("Bob", "Cecil", "Cecil", "David", "David", "Esmeralda", "Bob", "Alice",
     "Cecil", "David"),
  to = c("Alice", "Bob", "Alice", "Alice", "Bob", "Alice", "Bob", "Alice", "Cecil",
     "David"),
  friendship = c(4, 5, 5, 2, 1, 1, 2, 1, 3, 4),
  type = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2)
)
Network(relations, actors)
Network(relations, actors, theme = "theme_blank", theme_args = list(add_coord = FALSE))
Network(relations, actors, link_weight_by = "friendship", node_size_by = "age",
 link_weight_name = "FRIENDSHIP", node_fill_by = "gender", link_color_by = "to",
 link_type_by = "type", node_color_by = "black", layout = "circle", link_curvature = 0.2)
Network(relations, actors, layout = "tree", directed = FALSE, cluster = "fast_greedy",
 add_mark = TRUE)
Network(relations, actors, split_by = "type")
```

---

palette_list *A list of palettes for use in data visualization*

Description

A list of palettes for use in data visualization

Examples

```
## Not run:
if (interactive()) {
  library(stringr)
  library(RColorBrewer)
  library(Redmonder)
  library(rcartocolor)
  library(nord)
  library(viridis)
  library(pals)
  library(dichromat)
  library(jcolors)
  library(scales)
  syspals <- utils::getFromNamespace("syspals", "pals")
  brewer.pal.info <- RColorBrewer::brewer.pal.info
  ggsci_db <- utils::getFromNamespace("ggsci_db", "ggsci")
  redmonder.pal.info <- Redmonder::redmonder.pal.info
  metacartocolors <- rcartocolor::metacartocolors
  rownames(metacartocolors) <- metacartocolors$Name
  nord_palettes <- nord::nord_palettes
  viridis_names <- c("magma", "inferno", "plasma", "viridis", "cividis", "rocket",
     "mako", "turbo")
  viridis_palettes <- lapply(setNames(viridis_names, viridis_names),
     function(x) viridis::viridis(100, option = x))
  ocean_names <- names(syspals)[grep("ocean", names(syspals))]
  ocean_palettes <- syspals[ocean_names]
  dichromat_palettes <- dichromat::colorschemes
  jcolors_names <- paste0("jcolors-", c("default", "pal2", "pal3", "pal4", "pal5",
     "pal6", "pal7", "pal8", "pal9", "pal10", "pal11", "pal12", "rainbow"))
  custom_names <- c("jet", "simspec", "GdRd")
  custom_palettes <- list(
    oompaBase::jetColors(N = 100),
    c("#c22b86", "#f769a1", "#fcc5c1", "#253777", "#1d92c0", "#9ec9e1", "#015b33",
        "#42aa5e", "#d9f0a2", "#E66F00", "#f18c28", "#FFBB61"),
    c("gold", "red3")
  )
  names(custom_palettes) <- custom_names
  seurat_discrete_palettes <- list(
     alphabet = c(
       "#F0A0FF", "#0075DC", "#993F00", "#4C005C", "#191919", "#005C31",
       "#2BCE48", "#FFCC99", "#808080", "#94FFB5", "#8F7C00", "#9DCC00",
       "#C20088", "#003380", "#FFA405", "#FFA8BB", "#426600", "#FF0010",
       "#5EF1F2", "#00998F", "#E0FF66", "#740AFF", "#990000", "#FFFF80",
       "#FFE100", "#FF5005"
     ),
     alphabet2 = c(
       "#AA0DFE", "#3283FE", "#85660D", "#782AB6", "#565656", "#1C8356",
       "#16FF32", "#F7E1A0", "#E2E2E2", "#1CBE4F", "#C4451C", "#DEA0FD",
       "#FE00FA", "#325A9B", "#FEAF16", "#F8A19F", "#90AD1C", "#F6222E",
       "#1CFFCE", "#2ED9FF", "#B10DA1", "#C075A6", "#FC1CBF", "#B00068",
       "#FBE426", "#FA0087"
     ),
```

```
    glasbey = c(
      "#0000FF", "#FF0000", "#00FF00", "#000033", "#FF00B6", "#005300",
      "#FFD300", "#009FFF", "#9A4D42", "#00FFBE", "#783FC1", "#1F9698",
      "#FFACFD", "#B1CC71", "#F1085C", "#FE8F42", "#DD00FF", "#201A01",
      "#720055", "#766C95", "#02AD24", "#C8FF00", "#886C00", "#FFB79F",
      "#858567", "#A10300", "#14F9FF", "#00479E", "#DC5E93", "#93D4FF",
      "#004CFF", "#F2F318"
    ),
    polychrome = c(
      "#5A5156", "#E4E1E3", "#F6222E", "#FE00FA", "#16FF32", "#3283FE",
      "#FEAF16", "#B00068", "#1CFFCE", "#90AD1C", "#2ED9FF", "#DEA0FD",
      "#AA0DFE", "#F8A19F", "#325A9B", "#C4451C", "#1C8356", "#85660D",
      "#B10DA1", "#FBE426", "#1CBE4F", "#FA0087", "#FC1CBF", "#F7E1A0",
      "#C075A6", "#782AB6", "#AAF400", "#BDCDFF", "#822E1C", "#B5EFB5",
      "#7ED7D1", "#1C7F93", "#D85FF7", "#683B79", "#66B0FF", "#3B00FB"
    ),
    stepped = c(
      "#990F26", "#B33E52", "#CC7A88", "#E6B8BF", "#99600F", "#B3823E",
      "#CCAA7A", "#E6D2B8", "#54990F", "#78B33E", "#A3CC7A", "#CFE6B8",
      "#0F8299", "#3E9FB3", "#7ABECC", "#B8DEE6", "#3D0F99", "#653EB3",
      "#967ACC", "#C7B8E6", "#333333", "#666666", "#999999", "#CCCCCC"
    ),
    parade = c(
      '#ff6969', '#9b37ff', '#cd3737', '#69cdff', '#ffff69', '#69cdcd',
      '#9b379b', '#3737cd', '#ffff9b', '#cdff69', '#ff9b37', '#37ffff',
      '#9b69ff', '#37cd69', '#ff3769', '#ff3737', '#37ff9b', '#cdcd37',
      '#3769cd', '#37cdff', '#9b3737', '#ff699b', '#9b9bff', '#cd9b37',
      '#69ff37', '#cd3769', '#cd69cd', '#cd6937', '#3737ff', '#cdcd69',
      '#ff9b69', '#cd37cd', '#9bff37', '#cd379b', '#cd6969', '#69ff9b',
      '#ff379b', '#9bff9b', '#6937ff', '#69cd37', '#cdff37', '#9bff69',
      '#9b37cd', '#ff37ff', '#ff37cd', '#ffff37', '#37cd9b', '#379bff',
      '#ffcd37', '#379b37', '#ff9bff', '#379b9b', '#69ffcd', '#379bcd',
      '#ff69ff', '#ff9b9b', '#37ff69', '#ff6937', '#6969ff', '#699bff',
      '#ffcd69', '#69ffff', '#37ff37', '#6937cd', '#37cd37', '#3769ff',
      '#cd69ff', '#6969cd', '#9bcd37', '#69ff69', '#37cdcd', '#cd37ff',
      '#37379b', '#37ffcd', '#69cd69'
    )
)
seurat_continuous_palettes <- list(
  seurat.16 = hue_pal()(16),
  seurat.32 = hue_pal()(32),
  seurat.64 = hue_pal()(64)
)

palette_list <- list()
all_colors <- c(
  rownames(brewer.pal.info), names(ggsci_db), rownames(redmonder.pal.info),
  rownames(metacartocolors), names(nord_palettes), names(viridis_palettes),
  ocean_names, names(dichromat_palettes), jcolors_names, names(seurat_palettes),
  names(seurat_continuous_palettes), custom_names
)
for (pal in all_colors) {
  if (!pal %in% all_colors) {
```

```
  stop(paste0("Invalid pal Must be one of ", paste0(all_colors, collapse = ",")))
}
if (pal %in% rownames(brewer.pal.info)) {
  pal_n <- brewer.pal.info[pal, "maxcolors"]
  pal_category <- brewer.pal.info[pal, "category"]
  if (pal_category == "div") {
    palcolor <- rev(brewer.pal(name = pal, n = pal_n))
  } else {
    if (pal == "Paired") {
      palcolor <- brewer.pal(12, "Paired")[c(1:4, 7, 8, 5, 6, 9, 10, 11, 12)]
    } else {
      palcolor <- brewer.pal(name = pal, n = pal_n)
    }
  }
  if (pal_category == "qual") {
    attr(palcolor, "type") <- "discrete"
  } else {
    attr(palcolor, "type") <- "continuous"
  }
} else if (pal %in% names(ggsci_db)) {
  if (pal %in% c("d3", "uchicago", "material")) {
    for (subpal in names(ggsci_db[[pal]])) {
      palcolor <- ggsci_db[[pal]][[subpal]]
      if (pal == "material") {
        attr(palcolor, "type") <- "continuous"
      } else {
        attr(palcolor, "type") <- "discrete"
      }
      palette_list[[paste0(pal, "-", subpal)]] <- palcolor
    }
    next
  } else {
    palcolor <- ggsci_db[[pal]][[1]]
    if (pal == "gsea") {
      attr(palcolor, "type") <- "continuous"
    } else {
      attr(palcolor, "type") <- "discrete"
    }
  }
} else if (pal %in% rownames(redmonder.pal.info)) {
  pal_n <- redmonder.pal.info[pal, "maxcolors"]
  pal_category <- redmonder.pal.info[pal, "category"]
  if (pal_category == "div") {
    palcolor <- rev(redmonder.pal(name = pal, n = pal_n))
  } else {
    palcolor <- redmonder.pal(name = pal, n = pal_n)
  }
  if (pal_category == "qual") {
    attr(palcolor, "type") <- "discrete"
  } else {
    attr(palcolor, "type") <- "continuous"
  }
} else if (pal %in% rownames(metacartocolors)) {
```

```
      pal_n <- metacartocolors[pal, "Max_n"]
      palcolor <- carto_pal(name = pal, n = pal_n)
      if (pal_category == "qualitative") {
        attr(palcolor, "type") <- "discrete"
      } else {
        attr(palcolor, "type") <- "continuous"
      }
    } else if (pal %in% names(nord_palettes)) {
      palcolor <- nord_palettes[[pal]]
      attr(palcolor, "type") <- "discrete"
    } else if (pal %in% names(viridis_palettes)) {
      palcolor <- viridis_palettes[[pal]]
      attr(palcolor, "type") <- "continuous"
    } else if (pal %in% names(ocean_palettes)) {
      palcolor <- ocean_palettes[[pal]]
      attr(palcolor, "type") <- "continuous"
    } else if (pal %in% names(dichromat_palettes)) {
      palcolor <- dichromat_palettes[[pal]]
      if (pal %in% c("Categorical.12", "SteppedSequential.5")) {
        attr(palcolor, "type") <- "discrete"
      } else {
        attr(palcolor, "type") <- "continuous"
      }
    } else if (pal %in% jcolors_names) {
      palcolor <- jcolors(palette = gsub("jcolors-", "", pal))
      if (pal %in% paste0("jcolors-", c("pal10", "pal11", "pal12", "rainbow"))) {
        attr(palcolor, "type") <- "continuous"
      } else {
        attr(palcolor, "type") <- "discrete"
      }
    } else if (pal %in% custom_names) {
      palcolor <- custom_palettes[[pal]]
      if (pal %in% c("jet")) {
        attr(palcolor, "type") <- "continuous"
      } else {
        attr(palcolor, "type") <- "discrete"
      }
    } else if (pal %in% names(seurat_discrete_palettes)) {
      palcolor <- seurat_discrete_palettes[[pal]]
      attr(palcolor, "type") <- "discrete"
    } else if (pal %in% names(seurat_continuous_palettes)) {
      palcolor <- seurat_continuous_palettes[[pal]]
      attr(palcolor, "type") <- "continuous"
    }
    palette_list[[pal]] <- palcolor
  }
}

## End(Not run)
```

---

palette_this *Color palettes collected in plotthis.*

Description

Color palettes collected in plotthis.

Usage

```
palette_this(
  x,
  n = 100,
  palette = "Paired",
  palcolor = NULL,
  type = "auto",
  keep_names = TRUE,
  alpha = 1,
  matched = FALSE,
  reverse = FALSE,
  NA_keep = FALSE,
  NA_color = "grey80",
  transparent = TRUE
)
```

Arguments

x	A vector of character/factor or numeric values. If missing, numeric values 1:n will be used as x.
n	The number of colors to return for numeric values.
palette	Palette name. All available palette names can be queried with `show_palettes()`.
palcolor	Custom colors used to create a color palette.
type	Type of `x`. Can be one of "auto", "discrete" or "continuous". The default is "auto", which automatically detects if `x` is a numeric value.
keep_names	Whether to keep the names of the color vector.
alpha	The alpha value of the colors. Default is 1.
matched	If `TRUE`, will return a color vector of the same length as `x`.
reverse	Whether to invert the colors.
NA_keep	Whether to keep the color assignment to NA in `x`.
NA_color	Color assigned to NA if NA_keep is `TRUE`.
transparent	Whether to make the colors transparent when alpha < 1. When `TRUE`, `ggplot2::alpha()` is used to make the colors transparent. Otherwise, `adjcolors` is used to adjust the colors based on the alpha. The color will be not be actually transparent. For example, `ggplot2::alpha("red", 0.5) == "#FF000080"`; while `adjcolors("red", 0.5) == "#FF8080"`.

Value

A vector of colors.

PieChart *Pie Chart*

Description

Pie chart to illustrate numerical proportion of each group.

Usage

```
PieChart(
  data,
  x,
  y = NULL,
  label = y,
  split_by = NULL,
  split_by_sep = "_",
  clockwise = TRUE,
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)
```

Arguments

data	A data frame.

x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string of the column name to plot on the y-axis. A numeric column is expected. If NULL, the count of each x column will be used.
label	Which column to use as the label. NULL means no label.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
clockwise	A logical value to draw the pie chart clockwise or not.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
keep_empty	A logical value indicating whether to keep empty groups. If FALSE, empty groups will be removed.

combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
seed	The random seed to use. Default is 8525.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- data.frame(
   x = c("A", "B", "C", "D", "E", "F", "G", "H"),
   y = c(10, 8, 16, 4, 6, 12, 14, 2),
   group = c("G1", "G1", "G2", "G2", "G3", "G3", "G4", "G4"),
   facet = c("F1", "F2", "F3", "F4", "F1", "F2", "F3", "F4")
)

PieChart(data, x = "x", y = "y")
PieChart(data, x = "x", y = "y", clockwise = FALSE)
PieChart(data, x = "x", y = "y", label = "group")
PieChart(data, x = "x", y = "y", facet_by = "facet")
PieChart(data, x = "x", y = "y", split_by = "group")
PieChart(data, x = "x", y = "y", split_by = "group",
         palette = list(G1 = "Reds", G2 = "Blues", G3 = "Greens", G4 = "Purp"))

# y from count
PieChart(data, x = "group")
```

PrepareEnrichrResult *Enrichment Map/Network*

Description

PrepareEnrichrResult is a function to process the enrichment results from Enrichr. EnrichMap is a function to plot the enrichment map. EnrichNetwork is a function to plot the enrichment network.

Usage

```
PrepareEnrichrResult(data, n_input = NULL)

EnrichMap(
  data,
  split_by = NULL,
  split_by_sep = "_",
```

```
  top_term = 10,
  metric = "p.adjust",
  layout = "fr",
  minchar = 2,
  cluster = "fast_greedy",
  show_keyword = FALSE,
  nlabel = 4,
  character_width = 50,
  mark = "ellipse",
  label = c("term", "feature"),
  labelsize = 5,
  expand = c(0.4, 0.4),
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)

EnrichNetwork(
  data,
  split_by = NULL,
  split_by_sep = "_",
  top_term = 10,
  metric = "p.adjust",
  character_width = 50,
  layout = "fr",
  layoutadjust = TRUE,
  adjscale = 60,
  adjiter = 100,
  blendmode = "blend",
  labelsize = 5,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
```

```
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame containing the data to be plotted. It should be in the format of clusterProfiler enrichment result, which includes the columns: ID, Description, GeneRatio, BgRatio, pvalue, p.adjust, qvalue, geneID and Count. • The ID, qvalue and Count columns are optional. • The Description is the description of the term. • The GeneRatio is the number of genes in the term divided by the total number of genes in the input list. • The BgRatio is the number of genes in the term divided by the total number of genes in the background list (all terms). • The Count column, if given, should be the same as the first number in GeneRatio. If you have enrichment results from multiple databases, you can combine them into one data frame and add a column (e.g. Database) to indicate the database. You can plot them in a single plot using the split_by argument (e.g. split_by = "Database").
n_input	An integer specifying the number of input genes. Enrichr result doesn't ship with the number of input genes. You can either provide the number directly or we will infer it. See details.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
top_term	An integer specifying the number of top terms to show.
metric	A character string specifying the metric to use for the size of the nodes. It is also used to order the terms when selected the top terms. Either "pvalue" or "p.adjust". The default is "p.adjust".
layout	A character string specifying the layout of the graph. Either "circle", "tree", "grid" or other layout functions in igraph.

`minchar` An integer specifying the minimum number of characters to show in the keyword.

`cluster` A character string specifying the clustering method. Either "fast_greedy", "walktrap", "edge_betweenness", "infomap" or other clustering functions in `igraph`.

`show_keyword` A logical value specifying whether to show the keyword instead of Description/Term in the plot.

`nlabel` An integer specifying the number of labels to show in each cluster.

`character_width`
The width of the characters used to wrap the keyword.

`mark` A character string specifying the mark to use for the nodes. Either "ellipse", "rect", "circle", "text" or other mark functions in `ggforce`.

`label` A character string specifying the label to show in the legend. Either "term" or "feature". The default is "term".

`labelsize` A numeric value specifying the size of the label.

`expand` The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html

`theme` A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

`theme_args` A list of arguments to pass to the theme function.

`palette` A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.

`palcolor` A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

`alpha` A numeric value specifying the transparency of the plot.

`aspect.ratio` A numeric value specifying the aspect ratio of the plot.

`legend.position`
A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

`legend.direction`
A character string specifying the direction of the legend.

`title` A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.
layoutadjust	A logical value specifying whether to adjust the layout of the network.
adjscale	A numeric value specifying the scale of the adjustment.
adjiter	A numeric value specifying the number of iterations for the adjustment.
blendmode	A character string specifying the blend mode of the colors. Either "blend", "average", "multiply" and "screen".

Details

In order to use the EnrichMap and EnrichNetwork functions and other visualization functions in plotthis, the enrichment results from Enrichr need to be processed by the PrepareEnrichrResult function. The following columns are renamed:

- Term -> Description
- Genes -> geneID (separated replaced by /)
- P.value -> pvalue
- Adjusted.P.value -> p.adjust Additionally, GeneRatio and BgRatio columns are inferred. From enrichr's documentation, the oddsRatio is defined as: oddsRatio = (A * (D - B - C + A) / max((B - A) * (C - where A is the overlapping genes; B is the total genes in the gene set; C (n_input) is the genes in input list; D is the total genes in the background. D is not provided by Enrichr. To infer it, D = oddsRatio * max((B - A) * (C - A), 1) / A + B + C - A.
- Overlap = A / B (from Enrichr)
- GeneRatio = A / C (from ClusterProfiler)
- BgRatio = B / D (from ClusterProfiler) C (n_input), if not provided, will be inferred when D for all terms are equal. When starting inferrence, the minimum value to try will be unique genes in data$Genes/data$geneID.

Value

A data frame that can be used in EnrichMap.

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data(enrich_example)
EnrichMap(enrich_example)
EnrichMap(enrich_example, label = "feature")
EnrichMap(enrich_example, show_keyword = TRUE, label = "term")
EnrichMap(enrich_example, show_keyword = TRUE, label = "feature")

data(enrich_multidb_example)
EnrichMap(enrich_multidb_example, split_by = "Database")
EnrichMap(enrich_multidb_example, split_by = "Database",
          palette = list(DB1 = "Paired", DB2 = "Set1"))
EnrichNetwork(enrich_example, top_term = 5)
```

PrepareFGSEAResult *GSEA plots*

Description

- `GSEASummaryPlot` is used to plot a summary of the results of a GSEA analysis.
- `GSEAPlot` is used to plot the results of a GSEA analysis.
- `PrepareFGSEA` is used to prepare the result of GSEA from the `fgsea` package for plotting.

Usage

```
PrepareFGSEAResult(data, gene_ranks, gene_sets)

GSEASummaryPlot(
  data,
  gene_ranks = "@gene_ranks",
  gene_sets = "@gene_sets",
  top_term = 10,
  metric = "p.adjust",
  cutoff = 0.05,
  character_width = 50,
  line_plot_size = 0.25,
  metric_name = paste0("-log10(", metric, ")"),
  nonsig_name = "Non-significant",
  linewidth = 0.2,
  line_by = c("prerank", "running_score"),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  alpha = 0.6,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
```

```
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  seed = 8525,
  ...
)

GSEAPlot(
  data,
  gene_ranks = "@gene_ranks",
  gene_sets = "@gene_sets",
  sample_coregenes = FALSE,
  line_width = 1.5,
  line_alpha = 1,
  line_color = "#6BB82D",
  n_coregenes = 10,
  genes_label = NULL,
  label_fg = "black",
  label_bg = "white",
  label_bg_r = 0.1,
  label_size = 4,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)
```

Arguments

data	A data frame of GSEA results For example, from `DOSE::gseDO()`. Required columns are `ID`, `Description`, `NES`, `p.adjust`, `pvalue`. The `ID` column is used to match the gene sets.
gene_ranks	A numeric vector of gene ranks with genes as names The gene ranks are used to plot the gene sets. If `gene_ranks` is a character vector starting with `@`, the gene ranks will be taken from the attribute of `data`.
gene_sets	A list of gene sets, typically from a record of a GMT file The names of the list should match the `ID` column of `data`. If `gene_sets` is a character vector starting with `@`, the gene sets will be taken from the attribute of `data`. The GSEA plots will be plotted for each gene set. So, the number of plots will be the number of gene sets. If you only want to plot a subset of gene sets, you can subset the `gene_sets` before passing it to this function.

top_term An integer to select the top terms

metric The metric to use for the significance of the terms

cutoff The cutoff for the significance of the terms The terms will not be filtered with this cutoff; they are only filtered by the top_term ranked by the metric. The cutoff here is used to show the significance of the terms on the plot. For the terms that are not significant, the color will be grey.

character_width
The width of the characters in the y-axis

line_plot_size The size of the line plots

metric_name The name of the metric to show in the color bar

nonsig_name The name of the legend for the nonsignificant terms

linewidth The width of the lines in the line plots

line_by The method to calculate the line plots.

- prerank: Use the gene ranks as heights to plot the line plots.
- running_score: Use the running score to plot the line plots.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

alpha A numeric value specifying the transparency of the plot.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

seed The random seed to use. Default is 8525.

... Additional arguments.

sample_coregenes
A logical value to sample the core genes from the core_enrichment; if FALSE, the first n_coregenes will be used

line_width	The width of the line in the running score plot
line_alpha	The alpha of the line in the running score plot
line_color	The color of the line in the running score plot
n_coregenes	The number of core genes to label
genes_label	The genes to label. If set, n_coregenes will be ignored
label_fg	The color of the label text
label_bg	The background color of the label
label_bg_r	The radius of the background color of the label
label_size	The size of the label text
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.

Value

A data frame with the desired columns for plotting and the gene ranks and gene sets as attributes

Examples

```
data(gsea_example)
GSEASummaryPlot(gsea_example)
GSEASummaryPlot(gsea_example, line_by = "running_score")
GSEASummaryPlot(gsea_example, cutoff = 0.01)


GSEAPlot(gsea_example, gene_sets = attr(gsea_example, "gene_sets")[1])
GSEAPlot(gsea_example, gene_sets = attr(gsea_example, "gene_sets")[1:4])
```

PrepareUpsetData *Upset Plot*

Description

PrepareUpsetData is used to process the input data for Upset plot. UpsetPlot is used to plot the processed data.

Usage

```
PrepareUpsetData(
  data,
  in_form = NULL,
  group_by = NULL,
  group_by_sep = "_",
  id_by = NULL,
  specific = TRUE
)

UpsetPlot(
  data,
  in_form = NULL,
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  id_by = NULL,
  label = TRUE,
  label_fg = "black",
  label_size = NULL,
  label_bg = "white",
  label_bg_r = 0.1,
  palette = "material-indigo",
  palcolor = NULL,
  alpha = 1,
  specific = TRUE,
  theme = "theme_this",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  aspect.ratio = 0.6,
  legend.position = "right",
  legend.direction = "vertical",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)
```

Arguments

data	A data frame or a list or an UpsetPlotData object.
in_form	A character string indicating the datatype of the input data. Possible values are

"long", "wide", "list", "upset" or NULL. "long" indicates the data is in long format. "wide" indicates the data is in wide format. "list" indicates the data is a list. "upset" indicates the data is a UpsetPlotData object. NULL indicates the function will detect the datatype of the input data.

A long format data would look like:

```
group_by id_by
A        a1
A        a2
B        a1
B        a3
...
```

A wide format data would look like:

```
A    B
TRUE TRUE
TRUE FALSE
FALSE TRUE
...
```

A list format data would look like:

```
list(A = c("a1", "a2"), B = c("a1", "a3"))
```

An UpsetPlotData object is generated by PrepareUpsetData() would look like:

```
group_by
--------
list("A")  # a2
list("B")  # a3
list(c("A", "B"))  # a1
...
```

`group_by` A character string specifying the column name of the data frame to group the data.

`group_by_sep` A character string to concatenate the columns in `group_by`, if multiple columns are provided and the in_form is "long".

`id_by` A character string specifying the column name of the data frame to identify the instances. Required when `group_by` is a single column and data is a data frame.

`specific` A logical value to show the specific intersections only. ggVennDiagram, by default, only return the specific subsets of a region. However, sometimes, we want to show all the overlapping items for two or more sets. See https://github.com/gaospecial/ggVennDiagram/issues/64 for more details.

`split_by` The column(s) to split data by and plot separately.

`split_by_sep` The separator for multiple split_by columns. See `split_by`

`label` A logical value to show the labels on the bars.

`label_fg` A character string specifying the color of the label text.

`label_size` A numeric value specifying the size of the label text.

`label_bg` A character string specifying the background color of the label.

label_bg_r	A numeric value specifying the radius of the background of the label.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
seed	The random seed to use. Default is 8525.
...	Additional arguments.

Value

A UpsetPlotData object

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- list(
    A = 1:5,
    B = 2:6,
    C = 3:7,
    D = 4:8
)
UpsetPlot(data)
UpsetPlot(data, label = FALSE)
UpsetPlot(data, palette = "Reds", specific = FALSE)
```

PrepareVennData *VennDiagram*

Description

PrepareVennData is helper function to process the input data for Venn diagram.

Usage

```
PrepareVennData(
  data,
  in_form = NULL,
  group_by = NULL,
  group_by_sep = "_",
  id_by = NULL
)

VennDiagram(
  data,
  in_form = NULL,
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  id_by = NULL,
  label = "count",
  label_fg = "black",
  label_size = NULL,
  label_bg = "white",
  label_bg_r = 0.1,
  fill_mode = "count",
  fill_name = NULL,
  palette = ifelse(fill_mode == "set", "Paired", "Spectral"),
  palcolor = NULL,
  alpha = 1,
  theme = "theme_this",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
```

)

Arguments

data	A data frame or a list or a VennPlotData object.

in_form	A character string indicating the datatype of the input data. Possible values are "long", "wide", "list", "venn" or NULL. "long" indicates the data is in long format. "wide" indicates the data is in wide format. "list" indicates the data is a list. "venn" indicates the data is a VennPlotData object. NULL indicates the function will detect the datatype of the input data.
	A long format data would look like:
	`group_by id_by`
	`A a1`
	`A a2`
	`B a1`
	`B a3`
	`...`
	A wide format data would look like:
	`A B`
	`TRUE TRUE`
	`TRUE FALSE`
	`FALSE TRUE`
	`...`
	A list format data would look like:
	`list(A = c("a1", "a2"), B = c("a1", "a3"))`

group_by	A character string specifying the column name of the data frame to group the data.
group_by_sep	A character string to concatenate the columns in group_by, if multiple columns are provided and the in_form is "long".
id_by	A character string specifying the column name of the data frame to identify the instances. Required when group_by is a single column and data is a data frame.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
label	A character string specifying the label to show on the Venn diagram. Possible values are "count", "percent", "both", "none" and a function. "count" indicates the count of the intersection. "percent" indicates the percentage of the intersection. "both" indicates both the count and the percentage of the intersection. "none" indicates no label. If it is a function, if takes a data frame as input and returns a character vector as label. The data frame has columns "id", "X", "Y", "name", "item" and "count".
label_fg	A character string specifying the color of the label text.
label_size	A numeric value specifying the size of the label text.
label_bg	A character string specifying the background color of the label.

label_bg_r A numeric value specifying the radius of the background of the label.

fill_mode A character string specifying the fill mode of the Venn diagram. Possible values are "count", "set", "count_rev". "count" indicates the fill color is based on the count of the intersection. "set" indicates the fill color is based on the set of the intersection. "count_rev" indicates the fill color is based on the count of the intersection in reverse order. The palette will be continuous for "count" and "count_rev". The palette will be discrete for "set".

fill_name A character string to name the legend of colorbar.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

alpha A numeric value specifying the transparency of the plot.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

legend.position A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction A character string specifying the direction of the legend.

combine Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow A numeric value specifying the number of rows in the facet.

ncol A numeric value specifying the number of columns in the facet.

byrow A logical value indicating whether to fill the plots by row.

seed The random seed to use. Default is 8525.

... Additional arguments.

Value

A VennPlotData object

A combined ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
data = list(
    A = sort(sample(letters, 8)),
```

```
    B = sort(sample(letters, 8)),
    C = sort(sample(letters, 8)),
    D = sort(sample(letters, 8))
)

VennDiagram(data)
VennDiagram(data, fill_mode = "set")
VennDiagram(data, label = "both")
# label with a function
VennDiagram(data, label = function(df) df$name)
VennDiagram(data, palette = "material-indigo", alpha = 0.6)
```

RadarPlot *Radar plot / Spider plot*

Description

Create a radar plot or spider plot for a series of data. Radar plot uses circles as the plot grid and Spider plot uses polygons.

Usage

```
RadarPlot(
  data,
  x,
  x_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  y = NULL,
  group_name = NULL,
  scale_y = c("group", "global", "x", "none"),
  y_min = 0,
  y_max = NULL,
  y_nbreaks = 4,
  fill = TRUE,
  linewidth = 1,
  pt_size = 4,
  max_charwidth = 16,
  split_by = NULL,
  split_by_sep = "_",
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
```

```
  facet_nrow = NULL,
  facet_byrow = TRUE,
  alpha = 0.2,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)

SpiderPlot(
  data,
  x,
  x_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  y = NULL,
  group_name = NULL,
  scale_y = c("group", "global", "x", "none"),
  y_min = 0,
  y_max = NULL,
  y_nbreaks = 4,
  fill = TRUE,
  linewidth = 1,
  pt_size = 4,
  max_charwidth = 16,
  split_by = NULL,
  split_by_sep = "_",
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  alpha = 0.2,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
```

```
  subtitle = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
y	A character string specifying the column name of the data frame to plot for the y-axis.
group_name	A character string to name the legend of group.
scale_y	How should the y-axis be scaled? Default is "group". Other options are "global", "x" and "none". • If "group", the y-axis will be scaled to the fraction within each group. • If "global", the y-axis will be scaled to the fraction of the total. • If "x", the y-axis will be scaled to the fraction of the total within each x-axis group. • If "none", the y-axis will be scaled to the count of each x-axis group.
y_min	A numeric value to set the minimum value of the y-axis.
y_max	A numeric value to set the maximum value of the y-axis.
y_nbreaks	A numeric value to set the number of breaks in the y-axis.
fill	A logical value to fill the polygons with colors.
linewidth	A numeric value to set the width of the lines.
pt_size	A numeric value to set the size of the points.
max_charwidth	A numeric value to set the maximum character width for the x labels.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.

palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
alpha	A numeric value specifying the transparency of the plot.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
# use the count
data <- data.frame(
   x = c(rep("A", 2), rep("B", 3), rep("C", 3), rep("D", 4), rep("E", 5)),
   group = sample(paste0("G", 1:4), 17, replace = TRUE)
```

```
)
RadarPlot(data, x = "x")
RadarPlot(data, x = "x", scale_y = "none")
RadarPlot(data, x = "x", group_by = "group")
SpiderPlot(data, x = "x")
SpiderPlot(data, x = "x", group_by = "group")

# use the y value
data <- data.frame(
   x = rep(LETTERS[1:5], 2),
   y = c(1, 3, 6, 4, 2, 5, 7, 8, 9, 10),
   group = rep(c("G1", "G2"), each = 5)
)
RadarPlot(data, x = "x", y = "y", scale_y = "none", group_by = "group")
RadarPlot(data, x = "x", y = "y", facet_by = "group")
RadarPlot(data, x = "x", y = "y", split_by = "group")
RadarPlot(data, x = "x", y = "y", split_by = "group",
          palette = c(G1 = "Set1", G2 = "Paired"))
```

RarefactionPlot *RarefactionPlot*

Description

This function generates a rarefraction plot for a given dataset.

Usage

```
RarefactionPlot(
  data,
  type = 1,
  se = NULL,
  group_by = "group",
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 0.2,
  pt_size = 3,
  line_width = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
```

```
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
type	three types of plots: sample-size-based rarefaction/extrapolation curve (type = 1); sample completeness curve (type = 2); coverage-based rarefaction/extrapolation curve (type = 3).
se	a logical variable to display confidence interval around the estimated sampling curve. Default to NULL which means TRUE if the data has the lower and upper bounds.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	A character string indicating how to separate the group_by column if both "q" and "group" are used. for 'group_by'. Default to "_".
group_name	A character string indicating the name of the group, showing as the legend title.
split_by	A character string indicating how to split the data and plots Possible values are "q" and "group"
split_by_sep	The separator for multiple split_by columns. See split_by
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
pt_size	A numeric value specifying the size of the points.

line_width	A numeric value specifying the width of the lines.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
spider <- list(
   Girdled = c(46, 22, 17, 15, 15, 9, 8, 6, 6, 4, rep(2, 4), rep(1, 12)),
   Logged = c(88, 22, 16, 15, 13, 10, 8, 8, 7, 7, 7, 5, 4, 4, 4, 3, 3, 3, 3,
     2, 2, 2, 2, rep(1, 14))
)

RarefactionPlot(spider)
```

```
RarefactionPlot(spider, q = c(0, 1, 2), facet_by = "q")
RarefactionPlot(spider, q = c(0, 1, 2), split_by = "q")
RarefactionPlot(spider, q = c(0, 1, 2), split_by = "q",
                palette = c("0" = "Paired", "1" = "Set1", "2" = "Dark2"))
RarefactionPlot(spider, q = c(0, 1, 2), group_by = "q",
 facet_by = "group", palette = "Set1", type = 3)
```

RidgePlot *Ridge Plot*

Description

Ridge plot to illustrate the distribution of the data in different groups.

Usage

```
RidgePlot(
  data,
  x = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  scale = NULL,
  flip = FALSE,
  alpha = 1,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  x_text_angle = 90,
  keep_empty = FALSE,
  reverse = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = "none",
```

```
  legend.direction = "vertical",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
in_form	A character string specifying the form of the data. Default is "long".
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to name the legend of 'group_by', if 'legend.position' is not "none".
scale	A numeric value to scale the ridges. See also geom_density_ridges.
flip	A logical value. If TRUE, the plot will be flipped.
alpha	A numeric value specifying the transparency of the plot.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
x_text_angle	A numeric value specifying the angle of the x-axis text.

keep_empty — A logical value indicating whether to keep empty groups. If FALSE, empty groups will be removed.

reverse — A logical value. If TRUE, reverse the order of the groups on the y-axis.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

combine — Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

seed — The random seed to use. Default is 8525.

... — Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects. If no `split_by` is provided, a single plot (ggplot object) will be returned. If 'combine' is TRUE, a wrap_plots object will be returned. If 'combine' is FALSE, a list of ggplot objects will be returned.

Examples

```
set.seed(8525)
data <- data.frame(
   x = c(rnorm(250, -1), rnorm(250, 1)),
   group = rep(LETTERS[1:5], each = 100)
)
RidgePlot(data, x = "x", group_by = "group")
RidgePlot(data, x = "x", group_by = "group", reverse = TRUE)

# wide form
data_wide <- data.frame(
   A = rnorm(100),
```

```
  B = rnorm(100),
  C = rnorm(100),
  D = rnorm(100),
  E = rnorm(100),
  group = sample(letters[1:4], 100, replace = TRUE)
)
RidgePlot(data_wide, group_by = LETTERS[1:5], in_form = "wide")
RidgePlot(data_wide, group_by = LETTERS[1:5], in_form = "wide", facet_by = "group")
RidgePlot(data_wide, group_by = LETTERS[1:5], in_form = "wide", split_by = "group",
  palette = list(a = "Reds", b = "Blues", c = "Greens", d = "Purples"))
```

`RingPlot` *Ring Plot*

Description

A ring plot is like pie chart but with multiple rings.

Usage

```
RingPlot(
  data,
  x = NULL,
  y = NULL,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  label = NULL,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
```

```
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)
```

Arguments

data	A data frame.
x	A character vector specifying the column as the rings of the plot.
y	A character vector specifying the column as the y axis of the plot. Default is NULL, meaning the y axis is the count of the data.
group_by	A character vector specifying the column as the group_by of the plot. How the ring is divided.
group_by_sep	A character string to concatenate the columns in group_by, if multiple columns are provided.
group_name	A character string to specify the name of the group_by in the legend.
label	A logical value indicating whether to show the labels on the rings. The labels should be the values of group_by. Default is NULL, meaning no labels for one ring and showing the labels for multiple rings.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

alpha	A numeric value specifying the transparency of the plot.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
keep_empty	A logical value indicating whether to keep empty groups. If FALSE, empty groups will be removed.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
seed	The random seed to use. Default is 8525.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

See Also

PieChart

Examples

```
RingPlot(datasets::iris, group_by = "Species")

data <- data.frame(
  x = c("A", "B", "C", "A", "B", "C"),
  y = c(1, 2, 3, 4, 5, 6),
  group = c("a", "a", "a", "b", "b", "b")
)
RingPlot(data, x = "x", y = "y", group_by = "group")
RingPlot(datasets::mtcars, x = "cyl", group_by = "carb", facet_by = "vs")
RingPlot(datasets::mtcars, x = "cyl", group_by = "carb", split_by = "vs",
        palette = c("0" = "Set1", "1" = "Paired"))
```

SankeyPlot *Sankey / Alluvial Plot*

Description

A plot visualizing flow/movement/change from one state to another or one time to another. `AlluvialPlot` is an alias of `SankeyPlot`.

Usage

```
SankeyPlot(
  data,
  y = NULL,
  nodes_by,
  nodes_color = "grey30",
  links_by = NULL,
  links_by_sep = "_",
  links_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.6,
  nodes_label = FALSE,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  legend.box = "vertical",
  theme = "theme_this",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)

AlluvialPlot(
  data,
  y = NULL,
  nodes_by,
```

```
  nodes_color = "grey30",
  links_by = NULL,
  links_by_sep = "_",
  links_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.6,
  nodes_label = FALSE,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  legend.box = "vertical",
  theme = "theme_this",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)
```

Arguments

data	A data frame.
y	A character string specifying the column name of the data frame to plot for the y-axis.
nodes_by	A character vector of column names to define the nodes.
nodes_color	A character string to color the nodes.
links_by	A character vector of column names to define the links. If NULL, the links_by will be the first column in nodes_by.
links_by_sep	A character string to concatenate the columns in links_by, if multiple columns are provided.
links_name	A character string to name the legend of links.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
nodes_label	A logical value to show the labels on the nodes.
x_text_angle	A numeric value specifying the angle of the x-axis text.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
legend.box	A character string to specify the box of the legend, either "vertical" or "horizontal".
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
seed	The random seed to use. Default is 8525.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
data <- data.frame(
    nodes1 = sample(LETTERS[1:3], 10, replace = TRUE),
    nodes2 = sample(letters[1:3], 10, replace = TRUE),
    nodes3 = sample(LETTERS[4:6], 10, replace = TRUE),
    y = sample(1:5, 10, replace = TRUE)
)
```

```
SankeyPlot(data, nodes_by = c("nodes1", "nodes2", "nodes3"))
SankeyPlot(data, nodes_by = c("nodes1", "nodes2", "nodes3"), nodes_label = TRUE)
SankeyPlot(data, nodes_by = c("nodes1", "nodes2", "nodes3"), links_by = "y")
SankeyPlot(data, nodes_by = c("nodes1", "nodes2", "nodes3"), y = "y")
```

ScatterPlot *Scatter Plot*

Description

Scatter Plot

Usage

```
ScatterPlot(
  data,
  x,
  y,
  size_by = 2,
  size_name = NULL,
  color_by = NULL,
  color_name = NULL,
  color_reverse = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  shape = 21,
  alpha = ifelse(shape %in% 21:25, 0.65, 1),
  border_color = "black",
  highlight = NULL,
  highlight_shape = 16,
  highlight_size = 3,
  highlight_color = "red",
  highlight_alpha = 1,
  theme = "theme_this",
  theme_args = list(),
 palette = ifelse(!is.null(color_by) && !is.numeric(data[[color_by]]), "Paired",
    "Spectral"),
  palcolor = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
```

```
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
size_by	Which column to use as the size of the dots. It must be a numeric column. Or it can be a numeric value to specify the size of the dots.
size_name	A character vector specifying the name for the size legend.
color_by	Which column to use as the color of the dots. It could be a numeric column or a factor/character column. For shapes 21-25, the color is applied to the fill color.
color_name	A character vector specifying the name for the color legend.
color_reverse	A logical value indicating whether to reverse the color direction. Default is FALSE.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
shape	A numeric value specifying the shape of the points. Default is 21.
alpha	A numeric value specifying the transparency of the plot.
border_color	A character vector specifying the color for the border of the points. Or TRUE to use the fill color as the border color.
highlight	A vector of indexes or rownames to select the points to highlight. It could also be an expression (in string) to filter the data.
highlight_shape	A numeric value specifying the shape of the highlighted points. Default is 16.
highlight_size	A numeric value specifying the size of the highlighted points. Default is 3.
highlight_color	A character vector specifying the color of the highlighted points. Default is "red".
highlight_alpha	A numeric value specifying the transparency of the highlighted points. Default is 1.

theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
seed	The random seed to use. Default is 8525.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)

data <- data.frame(
   x = rnorm(20),
   y = rnorm(20),
   w = abs(rnorm(20)),
   t = sample(c("A", "B"), 20, replace = TRUE)
)
ScatterPlot(data, x = "x", y = "y")

# highlight points
ScatterPlot(data, x = "x", y = "y", highlight = 'x > 0')

# size_by is a numeric column
ScatterPlot(data, x = "x", y = "y", size_by = "w")

# color_by is a numeric column
ScatterPlot(data, x = "x", y = "y", color_by = "w")

# color_by is a factor/character column and set a border_color
ScatterPlot(data, x = "x", y = "y", size_by = "w", color_by = "t",
 border_color = "red")

# Same border_color as the fill color
ScatterPlot(data, x = "x", y = "y", size_by = "w", color_by = "t",
 border_color = TRUE)

# Shape doesn't have fill color
ScatterPlot(data, x = "x", y = "y", size_by = "w", color_by = "t",
 shape = 1, palette = "Set1")

# Change color per plot
ScatterPlot(data, x = "x", y = "y", split_by = "t",
            palcolor = list(A = "blue", B = "red"))
```

---

show_palettes *Show the color palettes*

Description

This function displays color palettes using ggplot2.

Usage

```
show_palettes(
  palettes = NULL,
  type = c("discrete", "continuous"),
  index = NULL,
```

```
  palette_names = NULL,
  return_names = TRUE,
  return_palettes = FALSE
)
```

Arguments

palettes	A list of color palettes. If NULL, uses default palettes.
type	A character vector specifying the type of palettes to include. Default is "discrete".
index	A numeric vector specifying the indices of the palettes to include. Default is NULL.
palette_names	A character vector specifying the names of the SCP palettes to include. Default is NULL.
return_names	A logical value indicating whether to return the names of the selected palettes. Default is TRUE.
return_palettes	A logical value indicating whether to return the colors of selected palettes. Default is FALSE.

Value

A list of palette names or a list of palettes.

See Also

palette_list

All available palettes

Examples

```
show_palettes(palettes = list(c("red", "blue", "green"), c("yellow", "purple", "orange")))
all_palettes <- show_palettes(return_palettes = TRUE)
names(all_palettes)
all_palettes[["simspec"]]
show_palettes(index = 1:10)
show_palettes(type = "discrete", index = 1:10)
show_palettes(type = "continuous", index = 1:10)
show_palettes(
    palette_names = c("Paired", "nejm", "simspec", "Spectral", "jet"),
    return_palettes = TRUE
)
```

`theme_blank` *Blank theme*

Description

This function creates a theme with all elements blank except for axis lines and labels. It can optionally add coordinate axes in the plot.

Usage

```
theme_blank(
  add_coord = TRUE,
  xlen_npc = 0.15,
  ylen_npc = 0.15,
  xlab = "",
  ylab = "",
  lab_size = 12,
  ...
)
```

Arguments

`add_coord`	Whether to add coordinate arrows. Default is `TRUE`.
`xlen_npc`	The length of the x-axis arrow in "npc".
`ylen_npc`	The length of the y-axis arrow in "npc".
`xlab`	x-axis label.
`ylab`	y-axis label.
`lab_size`	Label size.
`...`	Arguments passed to the `theme`.

Value

A ggplot2 theme.

Examples

```
library(ggplot2)
p <- ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(cyl))) +
    geom_point()
p + theme_blank()
p + theme_blank(xlab = "x-axis", ylab = "y-axis", lab_size = 16)
```

theme_this *A ggplot2 theme and palettes for plotthis Borrowed from the* theme_this *function in the* SCP *pipeline*

Description

A ggplot2 theme and palettes for plotthis Borrowed from the theme_this function in the SCP pipeline

Usage

```
theme_this(aspect.ratio = NULL, base_size = 12, font_family = NULL, ...)
```

Arguments

aspect.ratio	The aspect ratio of the plot
base_size	The base size of the text
font_family	The font family of the text
...	Other arguments for theme()

Value

A ggplot2 theme

See Also

https://github.com/zhanghao-njmu/SCP

TrendPlot *Trend plot*

Description

A trend plot is like an area plot but with gaps between the bars.

Usage

```
TrendPlot(
  data,
  x,
  y = NULL,
  x_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
```

```
  group_by_sep = "_",
  group_name = NULL,
  scale_y = FALSE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to name the legend of fill.
scale_y	A logical value to scale the y-axis by the total number in each x-axis group.

theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
x_text_angle	A numeric value specifying the angle of the x-axis text.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

See Also

AreaPlot

Examples

```
data <- data.frame(
    x = rep(c("A", "B", "C", "D"), 2),
    y = c(1, 3, 6, 4, 2, 5, 7, 8),
    group = rep(c("F1", "F2"), each = 4)
)
TrendPlot(data, x = "x", y = "y", group_by = "group")
TrendPlot(data, x = "x", y = "y", group_by = "group",
         scale_y = TRUE)
TrendPlot(data, x = "x", y = "y", split_by = "group")
TrendPlot(data, x = "x", y = "y", split_by = "group",
          palette = c(F1 = "Set1", F2 = "Paired"))
```

VolcanoPlot *Volcano plot*

Description

A volcano plot is a type of scatter plot that shows statistical significance (usually on the y-axis) versus magnitude of change (usually on the x-axis).

Usage

```
VolcanoPlot(
  data,
  x,
  y,
  ytrans = function(n) -log10(n),
  color_by = NULL,
  color_name = NULL,
  flip_negatives = FALSE,
  x_cutoff = NULL,
  y_cutoff = 0.05,
  split_by = NULL,
  split_by_sep = "_",
  x_cutoff_name = NULL,
  y_cutoff_name = NULL,
  x_cutoff_color = "red2",
  y_cutoff_color = "blue2",
  x_cutoff_linetype = "dashed",
  y_cutoff_linetype = "dashed",
  x_cutoff_linewidth = 0.5,
  y_cutoff_linewidth = 0.5,
  pt_size = 2,
```

```
  pt_alpha = 0.5,
  nlabel = 5,
  labels = NULL,
  label_size = 3,
  label_fg = "black",
  label_bg = "white",
  label_bg_r = 0.1,
  highlight = NULL,
  highlight_color = "red",
  highlight_size = 2,
  highlight_alpha = 1,
  highlight_stroke = 0.5,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
ytrans	A function to transform the y-axis values.
color_by	A character vector of column names to color the points by. If NULL, the points will be filled by the x and y cutoff value.
color_name	A character string to name the legend of color.

flip_negatives A logical value to flip the y-axis for negative x values.

x_cutoff A numeric value to set the x-axis cutoff. Both negative and positive of this value will be used.

y_cutoff A numeric value to set the y-axis cutoff. Note that the y-axis cutoff will be transformed by ytrans. So you should provide the original value.

split_by The column(s) to split data by and plot separately.

split_by_sep The separator for multiple split_by columns. See split_by

x_cutoff_name A character string to name the x-axis cutoff. If "none", the legend for the x-axis cutoff will not be shown.

y_cutoff_name A character string to name the y-axis cutoff. If "none", the legend for the y-axis cutoff will not be shown.

x_cutoff_color A character string to color the x-axis cutoff line.

y_cutoff_color A character string to color the y-axis cutoff line.

x_cutoff_linetype
A character string to set the x-axis cutoff line type.

y_cutoff_linetype
A character string to set the y-axis cutoff line type.

x_cutoff_linewidth
A numeric value to set the x-axis cutoff line size.

y_cutoff_linewidth
A numeric value to set the y-axis cutoff line size.

pt_size A numeric value to set the point size.

pt_alpha A numeric value to set the point transparency.

nlabel A numeric value to set the number of labels to show. The points will be ordered by the distance to the origin. Top nlabel points will be labeled.

labels A character vector of row names or indexes to label the points.

label_size A numeric value to set the label size.

label_fg A character string to set the label color.

label_bg A character string to set the label background color.

label_bg_r A numeric value specifying the radius of the background of the label.

highlight A character vector of row names or indexes to highlight the points.

highlight_color
A character string to set the highlight color.

highlight_size A numeric value to set the highlight size.

highlight_alpha
A numeric value to set the highlight transparency.

highlight_stroke
A numeric value to set the highlight stroke size.

facet_by A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.

Value

A list of ggplot objects or a wrap_plots object

Examples

```
set.seed(8525)
# Obtained by Seurat::FindMakers for the first cluster of pbmc_small
data <- data.frame(
   avg_log2FC = c(
     -3.69, -4.10, -2.68, -3.51, -3.09, -2.52, -3.53, -3.35, -2.82, -2.71, -3.16, -2.24,
     -5.62, -3.10, -3.42, -2.72, -3.23, -3.25, -4.68, 3.67, -2.66, 4.79, -2.99, 10.14,
     -1.78, -2.67, -2.26, -2.59, -3.39, 5.36, 4.56, 4.62, -2.94, -9.47, -9.12, -1.63,
     -2.77, 3.31, -1.53, -3.89, -4.21, 4.72, -2.98, -2.29, -1.41, -9.48, -4.30, 3.01,
     -1.19, -4.83, -1.35, -1.68, -1.63, -2.70, 3.86, 3.81, 7.23, -1.45, -0.92, -2.45,
     3.91, -4.45, -9.33, 3.56, 2.27, -1.60, -1.15, 11.40, -9.77, -8.32, 2.61, -1.25,
     -1.72, 10.61, 11.34, 10.02, 2.78, -3.48, -1.98, 5.86, 5.57, 4.57, 9.75, 9.97,
     10.90, 9.19, 2.93, 5.10, -1.52, -3.93, -1.95, -2.46, -0.64, 4.60, -1.82, -0.80,
     9.34, 7.51, 6.45, 5.23, 4.41, 3.60, -1.94, -1.15),
   p_val_adj = c(
     3.82e-09, 1.52e-07, 1.79e-07, 4.68e-07, 4.83e-07, 6.26e-07, 2.61e-06, 1.33e-05,
     1.79e-05, 3.71e-05, 5.21e-05, 5.36e-05, 5.83e-05, 6.66e-05, 8.22e-05, 2.89e-04,
     3.00e-04, 4.94e-04, 7.62e-04, 8.93e-04, 9.55e-04, 9.61e-04, 1.12e-03, 1.47e-03,
     1.66e-03, 1.95e-03, 2.06e-03, 3.01e-03, 3.26e-03, 4.35e-03, 4.85e-03, 5.12e-03,
     5.40e-03, 7.18e-03, 7.18e-03, 1.04e-02, 1.24e-02, 1.90e-02, 1.94e-02, 1.97e-02,
     2.09e-02, 2.13e-02, 2.25e-02, 2.61e-02, 3.18e-02, 3.27e-02, 3.69e-02, 3.80e-02,
     4.95e-02, 5.73e-02, 5.77e-02, 6.10e-02, 6.22e-02, 6.31e-02, 6.72e-02, 9.23e-02,
     9.85e-02, 1.06e-01, 1.07e-01, 1.11e-01, 1.31e-01, 1.38e-01, 1.40e-01, 1.43e-01,
     2.00e-01, 2.39e-01, 2.49e-01, 2.57e-01, 2.86e-01, 2.86e-01, 2.98e-01, 3.32e-01,
     4.15e-01, 4.91e-01, 4.91e-01, 4.91e-01, 5.97e-01, 7.11e-01, 7.59e-01, 8.38e-01,
     9.20e-01, 9.20e-01, 9.29e-01, 9.29e-01, 9.29e-01, 9.29e-01, 9.34e-01, 9.68e-01,
     1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00,
     1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00),
   gene = c(
     "HLA-DPB1", "LYZ", "HLA-DRA", "TYMP", "HLA-DPA1", "HLA-DRB1", "CST3", "HLA-DQB1",
     "HLA-DRB5", "LST1", "HLA-DQA1", "AIF1", "S100A8", "IFITM3", "HLA-DMB", "FCGRT",
     "SERPINA1", "IFI30", "S100A9", "CCL5", "GRN", "LCK", "HLA-DMA", "MS4A6A", "CTSS",
     "CFP", "FCN1", "BID", "CFD", "CD3D", "CD7", "CD3E", "LGALS2", "CD14", "SMCO4",
     "LINC00936", "HCK", "CTSW", "LGALS1", "HLA-DQA2", "LRRC25", "GZMM", "RNF130",
     "LGALS3", "S100A11", "C5AR1", "IL1B", "GZMA", "FCER1G", "MPEG1", "TYROBP", "TSPO",
     "GSTP1", "CTSB", "IL32", "CD247", "GNLY", "COTL1", "NFKBIA", "NUP214", "LAMP1",
     "FPR1", "CLEC10A", "CST7", "PRF1", "BLVRA", "PSAP", "GZMH", "EAF2", "ASGR1",
    "RARRES3", "SAT1", "LY86", "GP9", "TUBB1", "NGFRAP1", "XBP1", "SCO2", "RGS2", "GZMB",
     "HIST1H2AC", "KLRD1", "PGRMC1", "AKR1C3", "PTGDR", "IL2RB", "GYPC", "CCL4", "CD68",
     "FCER1A", "CD79B", "MS4A7", "CARD16", "ACAP1", "CD79A", "ANXA2", "TMEM40", "PF4",
     "GNG11", "CLU", "CD9", "FGFBP2", "TNFRSF1B", "IFI6"),
  pct_diff = c(
     -0.752, -0.457, -0.460, -0.671, -0.626, -0.701, -0.502, -0.619, -0.623, -0.598,
     -0.566, -0.626, -0.543, -0.566, -0.541, -0.542, -0.515, -0.489, -0.444, 0.428,
     -0.517, 0.461, -0.491, -0.410, -0.480, -0.491, -0.521, -0.491, -0.438, 0.411,
     0.411, 0.409, -0.438, -0.359, -0.359, -0.440, -0.386, 0.385, -0.332, -0.361, -0.361,
     0.364, -0.387, -0.415, -0.454, -0.308, -0.335, 0.364, -0.454, -0.309, -0.379, -0.427,
     -0.377, -0.389, 0.335, 0.315, 0.313, -0.284, -0.502, -0.309, 0.313, -0.284, -0.256,
     0.309, 0.313, -0.364, -0.406, 0.244, -0.231, -0.231, 0.281, -0.311, -0.312, 0.220,
     0.220, 0.220, 0.220, 0.261, -0.232, -0.367, 0.240, 0.218, 0.218, 0.195, 0.195, 0.195, 0.195,
     0.262, 0.218, -0.288, -0.207, -0.290, -0.233, -0.367, 0.217, -0.233, -0.403, 0.171,
     0.194, 0.194, 0.194, 0.194, 0.213, -0.235, -0.292),
```

```
  group = sample(LETTERS[1:2], 104, replace = TRUE)
)
rownames(data) <- data$gene

VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", color_by = "pct_diff",
   y_cutoff_name = "-log10(0.05)")
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   flip_negatives = TRUE)
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   flip_negatives = TRUE, facet_by = "group")
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   flip_negatives = TRUE, split_by = "group")
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   highlight = c("ANXA2", "TMEM40", "PF4", "GNG11", "CLU", "CD9", "FGFBP2",
   "TNFRSF1B", "IFI6"))
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", color_by = "pct_diff",
   y_cutoff_name = "-log10(0.05)", split_by = "group",
   palette = c(A = "Set1", B = "Dark2"))
```

WordCloudPlot *Word Cloud Plot*

Description

Word cloud plot to illustrate the count/frequency of words.

Usage

```
WordCloudPlot(
  data,
  word_by = NULL,
  sentence_by = NULL,
  count_by = NULL,
  score_by = NULL,
  count_name = NULL,
  score_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  words_excluded = plotthis::words_excluded,
  score_agg = mean,
  minchar = 2,
  word_size = c(2, 8),
  top_words = 100,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
```

```
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  palreverse = FALSE,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

data	A data frame.
word_by	A character string of the column name to use as the word. A character column is expected.
sentence_by	A character string of the column name to split the sentence. A character column is expected. Either word_by or sentence_by should be specified.
count_by	A character string of the column name for the count of the word/sentence. A numeric column is expected. If NULL, the count of the word/sentence will be used.
score_by	A character string of the column name for the score of the word/sentence. A numeric column is expected, used for the color of the word cloud. If NULL, the score will be set to 1.
count_name	A character string to name the legend of count.
score_name	A character string to name the legend of score.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
words_excluded	A character vector of words to exclude from the word cloud.
score_agg	A function to aggregate the scores. Default is mean.
minchar	A numeric value specifying the minimum number of characters for the word.
word_size	A numeric vector specifying the range of the word size.
top_words	A numeric value specifying the number of top words to show.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
palreverse	A logical value to reverse the palette colors.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- data.frame(
   word = c("apple", "banana", "cherry", "date", "elderberry"),
   count = c(10, 20, 30, 40, 50),
   score = c(1, 2, 3, 4, 5)
)
WordCloudPlot(data, word_by = "word", count_by = "count", score_by = "score")
```

---

words_excluded *Excluded words in keyword enrichment analysis and extraction*

Description

The variable "words_excluded" represents the words that are excluded during keyword enrichment analysis or keyword extraction process. These mainly include words that are excessively redundant or of little value.

Examples

```
## Not run:
if (interactive()) {
  words_excluded <- c(
    "the", "is", "and", "or", "a", "in", "on", "under", "between", "of", "through",
    "via", "along", "that", "for", "with", "within", "without", "cell", "cellular",
    "dna", "rna", "protein", "peptide", "amino", "acid", "development", "involved",
    "organization", "system", "regulation", "regulated", "positive", "negative",
    "response", "process", "processing", "small", "large", "change", "disease"
  )
}

## End(Not run)
```

Index