Package 'diceplot'

November 5, 2024

Title High Dimensional Categorical Data Visualization

Description Easy visualization for datasets with more than two categorical variables and additional continuous variables. 'diceplot' is particularly useful for exploring complex categorical data in the context of pathway analysis across multiple conditions. For a detailed documentation please visit <https://dice-and-domino-plot.readthedocs.io/en/latest/>.

Version 0.1.3

URL https://dice-and-domino-plot.readthedocs.io/en/latest/, https://github.com/maflot/Diceplot

BugReports https://github.com/maflot/Diceplot/issues

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr (>= 1.0.0), ggplot2 (>= 3.5.0), tidyr (>= 1.3.0), data.table (>= 1.14.8), cowplot, tibble, stats, rlang, RColorBrewer

NeedsCompilation no

Author Matthias Flotho [aut, cre] (<https://orcid.org/0009-0006-4374-0801>)

Maintainer Matthias Flotho <matthias.flotho@ccb.uni-saarland.de>

Repository CRAN

Date/Publication 2024-11-05 14:00:06 UTC

Contents

`calculate_dot_size` *Calculate Dynamic Dot Size*

Description

Calculates the dot size based on the number of variables.

Usage

```
calculate_dot_size(num_vars, max_size, min_size)
```

Arguments

`num_vars`	Number of variables.
`max_size`	Maximal dot size for the plot to scale the dot sizes.
`min_size`	Minimal dot size for the plot to scale the dot sizes.

Value

A numeric value representing the dot size.

`create_custom_legends` *Create Custom Legends*

Description

Creates custom legend plots for `cat_c` and `group`.

Usage

```
create_custom_legends(
  data,
  cat_c,
  group,
  cat_c_colors,
  group_colors,
  var_positions,
  num_vars,
  dot_size
)
```

Arguments

data	The original data frame.
cat_c	The name of the cat_c variable.
group	The name of the group variable.
cat_c_colors	A named vector of colors for cat_c.
group_colors	A named vector of colors for the group variable.
var_positions	Data frame with variable positions.
num_vars	Number of variables in cat_c.
dot_size	The size of the dots used in the plot.

Value

A combined ggplot object of the custom legends.

---

create_var_positions *Create Variable Positions*

Description

Generates a data frame containing variable names from cat_c_colors and corresponding x and y offsets based on the number of variables.

Usage

```
create_var_positions(cat_c_colors, num_vars)
```

Arguments

cat_c_colors	A named vector of colors for variables in category C. The names correspond to variable names.
num_vars	The number of variables. Supported values are "3", "4", "5", or "6".

Value

A data frame with columns:

var Factor of variable names from cat_c_colors.

x_offset Numeric x-axis offset for plotting.

y_offset Numeric y-axis offset for plotting.

Examples

```
library(dplyr)
cat_c_colors <- c("Var1" = "red", "Var2" = "blue", "Var3" = "green")
create_var_positions(cat_c_colors, 3)
```

dice_plot *Dice Plot Visualization*

Description

This function generates a custom plot based on three categorical variables and a group variable. It adapts to the number of unique categories in `cat_c` and allows customization of various plot aesthetics.

Usage

```
dice_plot(
  data,
  cat_a,
  cat_b,
  cat_c,
  group = NULL,
  group_alpha = 0.5,
  title = NULL,
  cat_c_colors = NULL,
  group_colors = NULL,
  custom_theme = theme_minimal(),
  max_dot_size = 5,
  min_dot_size = 2,
  legend_width = 0.25,
  legend_height = 0.5,
  base_width_per_cat_a = 0.5,
  base_height_per_cat_b = 0.3,
  reverse_ordering = FALSE,
  cat_b_order = NULL
)
```

Arguments

`data`	A data frame containing the categorical and group variables for plotting.
`cat_a`	A string representing the column name in `data` for the first categorical variable.
`cat_b`	A string representing the column name in `data` for the second categorical variable.
`cat_c`	A string representing the column name in `data` for the third categorical variable.
`group`	A string representing the column name in `data` for the grouping variable.
`group_alpha`	A numeric value for the transparency level of the group rectangles. Default is `0.5`.
`title`	An optional string for the plot title. Defaults to `NULL`.
`cat_c_colors`	A named vector of colors for `cat_c` categories or a string to chose a colorbrewer palette. Defaults to `NULL` using the first suitable colorbrewer palette to use.

group_colors	A named vector of colors for the group variableor a string to chose a colorbrewer palette. Defaults to NULL using the first suitable colorbrewer palette to use.
custom_theme	A ggplot2 theme for customizing the plot's appearance. Defaults to theme_minimal().
max_dot_size	Maximal dot size for the plot to scale the dot sizes.
min_dot_size	Minimal dot size for the plot to scale the dot sizes.
legend_width	Relative width of your legend. Default is 0.25.
legend_height	Relative width of your legend. Default is 0.5.
base_width_per_cat_a	Used for dynamically scaling the width. Default is 0.5.
base_height_per_cat_b	Used for dynamically scaling the height. Default is 0.3.
reverse_ordering	Should the cluster ordering be reversed?. Default is FALSE.
cat_b_order	Do you want to pass an explicit order?. Default is NULL.

Value

A ggplot object representing the dice plot.

---

domino_plot *Domino Plot Visualization*

Description

This function generates a plot to visualize gene expression levels for a given list of genes. The size of the dots can be customized, and the plot can be saved to an output file if specified.

Usage

```
domino_plot(
  data,
  gene_list,
  switch_axis = FALSE,
  min_dot_size = 1,
  max_dot_size = 5,
  spacing_factor = 3,
  var_id = "var",
  feature_col = "gene",
  celltype_col = "Celltype",
  contrast_col = "Contrast",
  contrast_levels = c("Clinical", "Pathological"),
  contrast_labels = c("Clinical", "Pathological"),
  logfc_col = "avg_log2FC",
  pval_col = "p_val_adj",
  logfc_limits = c(-1.5, 1.5),
```

```
  logfc_colors = c(low = "blue", mid = "white", high = "red"),
  color_scale_name = "Log2 Fold Change",
  size_scale_name = "-log10(adj. p-value)",
  axis_text_size = 8,
  aspect_ratio = NULL,
  base_width = 5,
  base_height = 4,
  output_file = NULL
)
```

Arguments

`data`	A data frame containing gene expression data.
`gene_list`	A character vector of gene names to include in the plot.
`switch_axis`	A logical value indicating whether to switch the x and y axes. Default is `FALSE`.
`min_dot_size`	A numeric value indicating the minimum dot size in the plot. Default is `1`.
`max_dot_size`	A numeric value indicating the maximum dot size in the plot. Default is `5`.
`spacing_factor`	A numeric value indicating the spacing between gene pairs. Default is `3`.
`var_id`	A string representing the column name in `data` for the variable identifier. Default is `"var"`.
`feature_col`	A string representing the column name in `data` for the feature variable (e.g., genes). Default is `"gene"`.
`celltype_col`	A string representing the column name in `data` for the cell type variable. Default is `"Celltype"`.
`contrast_col`	A string representing the column name in `data` for the contrast variable. Default is `"Contrast"`.
`contrast_levels`	A character vector specifying the levels of the contrast variable. Default is `c("Clinical", "Pathological")`.
`contrast_labels`	A character vector specifying the labels for the contrasts in the plot. Default is `c("Clinical", "Pathological")`.
`logfc_col`	A string representing the column name in `data` for the log fold change values. Default is `"avg_log2FC"`.
`pval_col`	A string representing the column name in `data` for the adjusted p-values. Default is `"p_val_adj"`.
`logfc_limits`	A numeric vector of length 2 specifying the limits for the log fold change color scale. Default is `c(-1.5, 1.5)`.
`logfc_colors`	A named vector specifying the colors for the low, mid, and high values in the color scale. Default is `c(low = "blue", mid = "white", high = "red")`.
`color_scale_name`	A string specifying the name of the color scale in the legend. Default is `"Log2 Fold Change"`.

size_scale_name	A string specifying the name of the size scale in the legend. Default is "-log10(adj.p-value)".
axis_text_size	A numeric value specifying the size of the axis text. Default is 8.
aspect_ratio	A numeric value specifying the aspect ratio of the plot. If NULL, it's calculated automatically. Default is NULL.
base_width	A numeric value specifying the base width for saving the plot. Default is 5.
base_height	A numeric value specifying the base height for saving the plot. Default is 4.
output_file	An optional string specifying the path to save the plot. If NULL, the plot is not saved. Default is NULL.

Value

A ggplot object representing the domino plot.

---

order_cat_b *Order Category B*

Description

Determines the ordering of category B based on the counts within each group, ordered by group and count.

Usage

```
order_cat_b(data, group, cat_b, group_colors, reverse_order = FALSE)
```

Arguments

data	A data frame containing the variables.
group	The name of the column representing the grouping variable.
cat_b	The name of the column representing category B.
group_colors	A named vector of colors for each group. The names correspond to group names.
reverse_order	Reverse the ordering? Default is FALSE.

Value

A vector of category B labels ordered according to group and count.

Examples

```
library(dplyr)
data <- data.frame(
  group = rep(c("G1", "G2"), each = 5),
  cat_b = sample(LETTERS[1:3], 10, replace = TRUE)
)
group_colors <- c("G1" = "red", "G2" = "blue")
order_cat_b(data, "group", "cat_b", group_colors)
```

perform_clustering *Perform Hierarchical Clustering on Category A*

Description

Performs hierarchical clustering on category A based on the binary presence of combinations of categories B and C.

Usage

```
perform_clustering(data, cat_a, cat_b, cat_c)
```

Arguments

data	A data frame containing the variables.
cat_a	The name of the column representing category A.
cat_b	The name of the column representing category B.
cat_c	The name of the column representing category C.

Value

A vector of category A labels ordered according to the hierarchical clustering.

Examples

```
library(dplyr)
library(tidyr)
library(tibble)
data <- data.frame(
  cat_a = rep(letters[1:5], each = 4),
  cat_b = rep(LETTERS[1:2], times = 10),
  cat_c = sample(c("Var1", "Var2", "Var3"), 20, replace = TRUE)
)
perform_clustering(data, "cat_a", "cat_b", "cat_c")
```

prepare_box_data *Prepare Box Data*

Description

Prepares data for plotting boxes by calculating box boundaries based on category positions.

Usage

```
prepare_box_data(data, cat_a, cat_b, group, cat_a_order, cat_b_order)
```

Arguments

data	A data frame containing the variables.
cat_a	The name of the column representing category A.
cat_b	The name of the column representing category B.
group	The name of the column representing the grouping variable.
cat_a_order	A vector specifying the order of category A.
cat_b_order	A vector specifying the order of category B.

Value

A data frame with box boundaries for plotting.

Examples

```
library(dplyr)
data <- data.frame(
  cat_a = rep(letters[1:3], each = 2),
  cat_b = rep(LETTERS[1:2], times = 3),
  group = rep(c("G1", "G2"), times = 3)
)
cat_a_order <- c("a", "b", "c")
cat_b_order <- c("A", "B")
prepare_box_data(data, "cat_a", "cat_b", "group", cat_a_order, cat_b_order)
```

---

prepare_plot_data *Prepare Plot Data*

Description

Prepares data for plotting by calculating positions based on provided variable positions and orders.

Usage

```
prepare_plot_data(
  data,
  cat_a,
  cat_b,
  cat_c,
  group,
  var_positions,
  cat_a_order,
  cat_b_order
)
```

Arguments

data	A data frame containing the variables.
cat_a	The name of the column representing category A.
cat_b	The name of the column representing category B.
cat_c	The name of the column representing category C.
group	The name of the column representing the grouping variable.
var_positions	A data frame with variable positions, typically output from create_var_positions.
cat_a_order	A vector specifying the order of category A.
cat_b_order	A vector specifying the order of category B.

Value

A data frame ready for plotting with added x_pos and y_pos columns.

Examples

```
library(dplyr)
data <- data.frame(
  cat_a = rep(letters[1:3], each = 4),
  cat_b = rep(LETTERS[1:2], times = 6),
  cat_c = rep(c("Var1", "Var2"), times = 6),
  group = rep(c("G1", "G2"), times = 6)
)
var_positions <- data.frame(
  var = c("Var1", "Var2"),
  x_offset = c(0.1, -0.1),
  y_offset = c(0.1, -0.1)
)
cat_a_order <- c("a", "b", "c")
cat_b_order <- c("A", "B")
prepare_plot_data(data, "cat_a", "cat_b", "cat_c", "group", var_positions, cat_a_order, cat_b_order)
```

Index