

# Package: diceplot (via r-universe)

October 22, 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/>](https://dice-and-domino-plot.readthedocs.io/en/latest/).

**Version** 0.1.0

**License** GPL (>= 2)

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**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

**Repository** <https://maflot.r-universe.dev>

**RemoteUrl** <https://github.com/maflot/diceplot>

**RemoteRef** HEAD

**RemoteSha** f3e76e6fa6de110ab0d1fa4657bc20562cbba938

## Contents

calculate_dot_size . . . . .	2
create_custom_legends . . . . .	2
create_var_positions . . . . .	3
dice_plot . . . . .	4
domino_plot . . . . .	5
order_cat_b . . . . .	7
perform_clustering . . . . .	8
prepare_box_data . . . . .	8
prepare_plot_data . . . . .	9

<b>Index</b>	<b>11</b>
--------------	-----------

---

calculate\_dot\_size     *Calculate Dynamic Dot Size*

---

**Description**

Calculates the dot size based on the number of variables.

**Usage**

```
calculate_dot_size(num_vars)
```

**Arguments**

num\_vars     Number of variables.

**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**

<code>data</code>	The original data frame.
<code>cat_c</code>	The name of the <code>cat_c</code> variable.
<code>group</code>	The name of the group variable.
<code>cat_c_colors</code>	A named vector of colors for <code>cat_c</code> .
<code>group_colors</code>	A named vector of colors for the group variable.
<code>var_positions</code>	Data frame with variable positions.
<code>num_vars</code>	Number of variables in <code>cat_c</code> .
<code>dot_size</code>	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**

<code>cat_c_colors</code>	A named vector of colors for variables in category C. The names correspond to variable names.
<code>num_vars</code>	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,  
  plot_path = NULL,  
  output_str = NULL,  
  group_alpha = 0.5,  
  title = NULL,  
  cat_c_colors = NULL,  
  group_colors = NULL,  
  format = ".pdf",  
  custom_theme = theme_minimal()  
)
```

### Arguments

<code>data</code>	A data frame containing the categorical and group variables for plotting.
<code>cat_a</code>	A string representing the column name in <code>data</code> for the first categorical variable.
<code>cat_b</code>	A string representing the column name in <code>data</code> for the second categorical variable.
<code>cat_c</code>	A string representing the column name in <code>data</code> for the third categorical variable.
<code>group</code>	A string representing the column name in <code>data</code> for the grouping variable.
<code>plot_path</code>	A string specifying the file path where the plot will be saved. Defaults to NULL (no file output).
<code>output_str</code>	An optional string to customize the filename if <code>plot_path</code> is specified. Defaults to NULL.
<code>group_alpha</code>	A numeric value for the transparency level of the group rectangles. Default is 0.5.
<code>title</code>	An optional string for the plot title. Defaults to NULL.
<code>cat_c_colors</code>	A named vector of colors for <code>cat_c</code> categories. Defaults to NULL (automatically generated).

group_colors	A named vector of colors for the group variable. Defaults to NULL (automatically generated).
format	A string specifying the output file format if saving the plot. Defaults to ".pdf".
custom_theme	A ggplot2 theme for customizing the plot's appearance. Defaults to theme_minimal().

**Value**

A ggplot object representing the dice plot.

---

domino_plot	<i>Domino Plot Visualization</i>
-------------	----------------------------------

---

**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**

<code>data</code>	A data frame containing gene expression data.
<code>gene_list</code>	A character vector of gene names to include in the plot.
<code>switch_axis</code>	A logical value indicating whether to switch the x and y axes. Default is FALSE.
<code>min_dot_size</code>	A numeric value indicating the minimum dot size in the plot. Default is 1.
<code>max_dot_size</code>	A numeric value indicating the maximum dot size in the plot. Default is 5.
<code>spacing_factor</code>	A numeric value indicating the spacing between gene pairs. Default is 3.
<code>var_id</code>	A string representing the column name in data for the variable identifier. Default is "var".
<code>feature_col</code>	A string representing the column name in data for the feature variable (e.g., genes). Default is "gene".
<code>celltype_col</code>	A string representing the column name in data for the cell type variable. Default is "Celltype".
<code>contrast_col</code>	A string representing the column name in data for the contrast variable. Default is "Contrast".
<code>contrast_levels</code>	A character vector specifying the levels of the contrast variable. Default is c("Clinical", "Pathological").
<code>contrast_labels</code>	A character vector specifying the labels for the contrasts in the plot. Default is c("Clinical", "Pathological").
<code>logfc_col</code>	A string representing the column name in data for the log fold change values. Default is "avg_log2FC".
<code>pval_col</code>	A string representing the column name in data for the adjusted p-values. Default is "p_val_adj".
<code>logfc_limits</code>	A numeric vector of length 2 specifying the limits for the log fold change color scale. Default is c(-1.5, 1.5).
<code>logfc_colors</code>	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").
<code>color_scale_name</code>	A string specifying the name of the color scale in the legend. Default is "Log2 Fold Change".
<code>size_scale_name</code>	A string specifying the name of the size scale in the legend. Default is "-log10(adj. p-value)".
<code>axis_text_size</code>	A numeric value specifying the size of the axis text. Default is 8.
<code>aspect_ratio</code>	A numeric value specifying the aspect ratio of the plot. If NULL, it's calculated automatically. Default is NULL.
<code>base_width</code>	A numeric value specifying the base width for saving the plot. Default is 5.
<code>base_height</code>	A numeric value specifying the base height for saving the plot. Default is 4.
<code>output_file</code>	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	<i>Order Category B</i>
-------------	-------------------------

---

**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)
```

**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.

**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**

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

`calculate_dot_size`, [2](#)  
`create_custom_legends`, [2](#)  
`create_var_positions`, [3](#)  
  
`dice_plot`, [4](#)  
`domino_plot`, [5](#)  
  
`order_cat_b`, [7](#)  
  
`perform_clustering`, [8](#)  
`prepare_box_data`, [8](#)  
`prepare_plot_data`, [9](#)