

Package ‘pcutils’

May 9, 2026

Type Package

Title Some Useful Functions for Statistics and Visualization

Version 0.2.8

Description Offers a range of utilities and functions for everyday programming tasks. 1.Data Manipulation. Such as grouping and merging, column splitting, and character expansion. 2.File Handling. Read and convert files in popular formats. 3.Plotting Assistance. Helpful utilities for generating color palettes, validating color formats, and adding transparency. 4.Statistical Analysis. Includes functions for pairwise comparisons and multiple testing corrections, enabling perform statistical analyses with ease. 5.Graph Plotting. Provides efficient tools for creating doughnut plot and multi-layered doughnut plot; Venn diagrams, including traditional Venn diagrams, upset plots, and flower plots; Simplified functions for creating stacked bar plots, or a box plot with alphabets group for multiple comparison group.

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 4.1.0)

Imports dplyr, magrittr, ggplot2, grid, stats, utils, grDevices, reshape2, scales, tools, tidyr, tibble, RColorBrewer, graphics

Suggests agricolae, clipr, rlang, BiocManager, ggpubr, kableExtra, htmlwidgets, pagedown, ggsci, readr, grImport2, rsvg, PMCMRplus, nortest, fitdistrplus, ggalluvial, gghalves, ggspatial, sf, magick, ggimage, ggpmisc, UpSetR, eulerr, plotrix, vegan, circlize, igraph, knitr, rmarkdown, plotly, htmltools, leaflet, relaimpo, snow, doSNOW, foreach, stringr, ggraph, ggrepel, treemap, voronoiTreemap, devtools, multcompView, rio, bookdown, sysfonts, showtext, jsonlite, httr, r.proxy, openssl, styler, lintr, aplot, ggbeeswarm, ggVennDiagram, gifski, ggnewscale, revtools

Config/Needs/website pkgdown, rabioco/rbitemplate

BugReports <https://github.com/Asa12138/pcutils/issues>

URL <https://github.com/Asa12138/pcutils>

Date/Publication 2025-03-27 06:10:02 UTC

NeedsCompilation no

Author Chen Peng [aut, cre] (ORCID: <<https://orcid.org/0000-0002-9449-7606>>)

Maintainer Chen Peng <pengchen2001@zju.edu.cn>

Repository CRAN

Contents

add_alpha	4
add_analysis	4
add_theme	5
change_fac_lev	5
check_directory_structure	6
china_map	7
copy_df	7
copy_vector	8
count2	8
dabiao	9
del_ps	10
df2distance	10
df2link	11
distance2df	11
download2	12
download_ncbi_genome_file	13
explode	14
fittest	15
generate_labels	15
get_cols	16
get_legend2	17
gghist	17
gghuan	18
gghuan2	19
ggmosaic	20
ggplot_lim	21
ggplot_translator	22
grepl.data.frame	23
group_box	23
group_test	25
gsub.data.frame	26
guolv	27
hebing	27
hebing2	28
how_to_set_font_for_plot	29
how_to_set_options	29
how_to_update_parameters	30
how_to_use_parallel	30
how_to_use_sbatch	31
igraph_translator	31

is.ggplot.color	32
legend_size	33
lib_ps	33
list_to_dataframe	34
little_guodong	34
lm_coefficients	34
make_gitbook	35
make_project	36
make_py_pkg	36
match_df	37
metadata	38
mmscale	38
multireg	39
multitest	39
my_cat	40
my_circle_packing	41
my_circo	42
my_lm	43
my_sunburst	44
my_treemap	44
my_voronoi_treemap	45
otutab	46
plot.coefficients	46
plotgif	47
plotpdf	47
prepare_package	48
pre_number_str	49
read.file	49
read_fasta	50
remove.outliers	50
rgb2code	51
rm_low	52
sample_map	52
sanxian	54
scale_color_pc	55
scale_fill_pc	55
search_browse	56
set_pcutils_config	57
show_pcutils_config	57
split_text	58
squash	58
stackplot	59
strsplit2	61
t2	62
taxonomy	62
tax_pie	63
tidai	63
trans	64

translator	65
trans_format	65
twotest	66
update_NEWS_md	67
update_param	67
venn	68
write_fasta	69

Index	70
--------------	-----------

add_alpha	<i>Add alpha for a Rcolor</i>
-----------	-------------------------------

Description

Add alpha for a Rcolor

Usage

```
add_alpha(color, alpha = 0.3)
```

Arguments

color	Rcolor
alpha	alpha, default 0.3

Value

8 hex color

Examples

```
add_alpha("red", 0.3)
```

add_analysis	<i>Add an analysis for a project</i>
--------------	--------------------------------------

Description

Add an analysis for a project

Usage

```
add_analysis(analysis_n, title = analysis_n, pro_dir = getwd())
```

Arguments

<code>analysis_n</code>	analysis name
<code>title</code>	Rmd file title
<code>pro_dir</code>	project directory, default is current directory

Value

No return value

<code>add_theme</code>	<i>Add a global gg_theme and colors for plots</i>
------------------------	---

Description

Add a global gg_theme and colors for plots

Usage

```
add_theme(set_theme = NULL)
```

Arguments

<code>set_theme</code>	your theme
------------------------	------------

Value

No return value

Examples

```
add_theme()
```

<code>change_fac_lev</code>	<i>Change factor levels</i>
-----------------------------	-----------------------------

Description

Change factor levels

Usage

```
change_fac_lev(x, levels = NULL, last = FALSE)
```

Arguments

x	vector
levels	custom levels
last	put the custom levels to the last

Value

factor

Examples

```
change_fac_lev(letters[1:5], levels = c("c", "a"))
```

check_directory_structure

Check if a directory structure matches the expected structure

Description

This function compares the actual directory structure with a predefined expected structure and returns TRUE if they match, otherwise FALSE. If verbose is TRUE, it prints detailed information about missing or extra files/directories.

Usage

```
check_directory_structure(
  root_path,
  expected_structure,
  only_missing = TRUE,
  verbose = FALSE
)
```

Arguments

root_path	A character string specifying the root directory to check.
expected_structure	A character vector specifying the expected directory structure. Each element should be a relative path (e.g., "data/raw").
only_missing	Only check the missing files/directories.
verbose	A logical value. If TRUE, prints detailed information; if FALSE, suppresses output.

Value

A logical value: TRUE if the directory structure matches the expected structure, otherwise FALSE.

china_map	<i>Plot china map</i>
-----------	-----------------------

Description

Plot china map

Usage

```
china_map(china_shp = NULL, download_dir = "pcutils_temp", text_param = NULL)
```

Arguments

china_shp	china.json file
download_dir	download_dir, "pcutils_temp"
text_param	parameters parse to geom_text

Value

a ggplot

copy_df	<i>Copy a data.frame</i>
---------	--------------------------

Description

Copy a data.frame

Usage

```
copy_df(df)
```

Arguments

df	a R data.frame object
----	-----------------------

Value

No return value

copy_vector	<i>Copy a vector</i>
-------------	----------------------

Description

Copy a vector

Usage

```
copy_vector(vec)
```

Arguments

vec a R vector object

Value

No return value

count2	<i>Like uniq -c in shell to count a vector</i>
--------	--

Description

Like uniq -c in shell to count a vector

Usage

```
count2(df)
```

Arguments

df two columns: first is type, second is number

Value

two columns: first is type, second is number

Examples

```
count2(data.frame(group = c("A", "A", "B", "C", "C", "A"), value = c(2, 2, 2, 1, 3, 1)))
```

dabiao *Print some message with =*

Description

Print some message with =

Usage

```
dabiao(  
    str = "",  
    ...,  
    n = 80,  
    char = "=",  
    mode = c("middle", "left", "right"),  
    print = FALSE  
)
```

Arguments

str	output strings
...	strings will be paste together
n	the number of output length
char	side chars default:=
mode	"middle", "left" or "right"
print	print or message?

Value

No return value

Examples

```
dabiao("Start running!")
```

del_ps	<i>Detach packages</i>
--------	------------------------

Description

Detach packages

Usage

```
del_ps(p_list, ..., origin = NULL)
```

Arguments

p_list	a vector of packages list
...	packages
origin	keep the original Namespace

Value

No return value

df2distance	<i>Convert Three-column Data to Distance Matrix</i>
-------------	---

Description

This function converts a data frame with three columns (from, to, count) into a distance matrix. The rows and columns of the matrix are all unique names from the 'from' and 'to' columns, and the matrix values are filled with counts.

Usage

```
df2distance(data)
```

Arguments

data	A data frame containing three columns: from, to, count.
------	---

Value

A distance matrix where rows and columns are all unique names from 'from' and 'to' columns.

Examples

```
data <- data.frame(
  from = c("A", "A", "B", "D"),
  to = c("B", "C", "A", "B"),
  count = c(1, 2, 3, 4)
)
df2distance(data)
```

df2link

df to link table

Description

df to link table

Usage

```
df2link(test, fun = sum)
```

Arguments

test	df with at least 3 columns
fun	function to summary the elements number, defalut: sum, you can choose mean.

Value

data.frame

Examples

```
data(otutab)
cbind(taxonomy, num = rowSums(otutab))[1:10, ] -> test
df2link(test)
```

distance2df

Convert a distance matrix to a data frame

Description

This function converts a distance matrix into a data frame with three columns: from, to, count. The rows and columns of the matrix are all unique names from the 'from' and 'to' columns,

Usage

```
distance2df(distance_matrix)
```

Arguments`distance_matrix`

A distance matrix where rows and columns are all unique names from 'from' and 'to' columns.

Value

A data frame containing three columns: from, to, count.

Examples

```
distance_matrix <- matrix(c(0, 1, 2, 3, 4, 5, 6, 7, 8), nrow = 3)
distance2df(distance_matrix)
```

`download2`*Download File*

Description

This function downloads a file from the provided URL and saves it to the specified location.

Usage

```
download2(url, file_path, timeout = 300, force = FALSE, proxy = FALSE, ...)
```

Arguments

<code>url</code>	The URL from which to download the file.
<code>file_path</code>	The full path to the file.
<code>timeout</code>	timeout, 300s
<code>force</code>	FALSE, if TRUE, overwrite existed file
<code>proxy</code>	use proxy, default is FALSE
<code>...</code>	add

Value

No value

`download_ncbi_genome_file`*Download genome files from NCBI based on accession number*

Description

This function downloads specific genomic files from NCBI's FTP server based on the provided accession number. It supports downloading different types of files, or the entire directory containing the files.

Usage

```
download_ncbi_genome_file(  
    accession,  
    out_dir = ".",  
    type = "gff",  
    file_suffix = NULL,  
    timeout = 300  
)
```

Arguments

<code>accession</code>	A character string representing the NCBI accession number (e.g., "GCF_001036115.1_ASM103611v1" or "GCF_001036115.1"). The accession can start with "GCF" or "GCA".
<code>out_dir</code>	A character string representing the directory where the downloaded files will be saved. Defaults to the current working directory (".").
<code>type</code>	A character string representing the type of file to download. Supported types are "all", "gff", "fna". If "all" is specified, the function will prompt the user to use command line tools to download the entire directory. Defaults to "gff".
<code>file_suffix</code>	A character string representing the specific file suffix to download. If specified, this will override the type parameter. Defaults to NULL.
<code>timeout</code>	A numeric value representing the maximum time in seconds to wait for the download. Defaults to 300.

Details

If the provided accession does not contain the version suffix (e.g., "GCF_001036115.1"), the function will query the NCBI FTP server to determine the full accession name.

When type is set to "all", the function cannot download the entire directory directly but provides a command line example for the user to download the directory using tools like `wget`.

Value

No value

Examples

```
## Not run:  
download_ncbi_genome_file("GCF_001036115.1", out_dir = "downloads", type = "gff")  
download_ncbi_genome_file("GCF_001036115.1", out_dir = "downloads", file_suffix = "_genomic.fna.gz")  
  
## End(Not run)
```

explode

Explode a data.frame if there are split charter in one column

Description

Explode a data.frame if there are split charter in one column

Usage

```
explode(df, column, split = ",")
```

Arguments

df	data.frame
column	column
split	split string

Value

data.frame

Examples

```
df <- data.frame(a = 1:2, b = c("a,b", "c"), c = 3:4)  
explode(df, "b", ",")
```

fittest	<i>Fit a distribution</i>
---------	---------------------------

Description

Fit a distribution

Usage

```
fittest(a)
```

Arguments

a	a numeric vector
---	------------------

Value

distribution

generate_labels	<i>Generate labels position</i>
-----------------	---------------------------------

Description

Generate labels position

Usage

```
generate_labels(
  labels = NULL,
  input = c(0, 0),
  nrows = NULL,
  ncols = NULL,
  x_offset = 0.3,
  y_offset = 0.15,
  just = 1
)
```

Arguments

labels	labels
input	c(0,0)
nrows	default: NULL
ncols	default: NULL
x_offset	0.3
y_offset	0.15
just	0~5

Value

matrix

Examples

```
library(ggplot2)
labels <- vapply(1:8, \(i)paste0(sample(LETTERS, 4), collapse = ""), character(1))
df <- data.frame(label = labels, generate_labels(labels))
ggplot(data = df) +
  geom_label(aes(x = X1, y = X2, label = label))
```

get_cols

Get n colors

Description

Get n colors

Usage

```
get_cols(n = 11, pal = NULL, n_break = 5)
```

Arguments

n	how many colors you need
pal	"col1", "col2", "col3"; or a vector of colors, you can get from: <code>RColorBrewer::brewer.pal(5, "Set2")</code> or <code>ggsci::pal_aaas()(5)</code>
n_break	default: 5

Value

a vector of n colors

Examples

```
get_cols(10, "col2") -> my_cols
scales::show_col(my_cols)

scales::show_col(get_cols(15, RColorBrewer::brewer.pal(5, "Set2")))
```

get_legend2	<i>Get a legend from a ggplot object</i>
-------------	--

Description

Get a legend from a ggplot object

Usage

```
get_legend2(plot, legend = NULL)
```

Arguments

plot	a ggplot object
legend	NULL, or position ("top")

Value

a grob object, or NULL if no legend found

Examples

```
library(ggplot2)
p <- ggplot(mtcars, aes(wt, mpg, color = mpg)) +
  geom_point()
legend <- get_legend2(p)
plot(legend)
```

gghist	<i>gg histogram</i>
--------	---------------------

Description

gg histogram

Usage

```
gghist(x, text_pos = c(0.8, 0.8), ...)
```

Arguments

x	vector
text_pos	text position, default is c(0.8, 0.8)
...	parameters parse to gghistogram

Value

ggplot

Examples

```
if (requireNamespace("ggpubr")) {
  gghist(rnorm(100))
}
```

gghuan

Plot a doughnut chart

Description

Plot a doughnut chart

Usage

```
gghuan(
  tab,
  reorder = TRUE,
  mode = "1",
  topN = 5,
  name = TRUE,
  percentage = TRUE,
  bar_params = NULL,
  text_params = NULL,
  text_params2 = NULL
)
```

Arguments

tab	two columns: first is type, second is number
reorder	reorder by number?
mode	plot style, 1~3
topN	plot how many top items
name	label the name
percentage	label the percentage
bar_params	parameters parse to <code>geom_rect</code> , for mode=1,3 or <code>geom_col</code> for mode=2.
text_params	parameters parse to <code>geom_text</code>
text_params2	parameters parse to <code>geom_text</code> , for name=TRUE & mode=1,3

Value

a ggplot

Examples

```

a <- data.frame(type = letters[1:6], num = c(1, 3, 3, 4, 5, 10))
gghuan(a) + scale_fill_pc()
gghuan(a,
  bar_params = list(col = "black"),
  text_params = list(col = "#b15928", size = 3),
  text_params2 = list(col = "#006d2c", size = 5)
) + scale_fill_pc()
gghuan(a, mode = 2) + scale_fill_pc()
gghuan(a, mode = 3) + scale_fill_pc()

```

gghuan2

*gghuan2 for multi-doughnut chart***Description**

gghuan2 for multi-doughnut chart

Usage

```

gghuan2(
  tab = NULL,
  huan_width = 1,
  circle_width = 1,
  space_width = 0.2,
  circle_label = NULL,
  pal = NULL,
  name = TRUE,
  percentage = FALSE,
  text_params = NULL,
  circle_label_params = NULL,
  bar_params = NULL
)

```

Arguments

tab	a dataframe with hierarchical structure
huan_width	the huan width (numeric vector)
circle_width	the center circle width
space_width	the space width between doughnuts (0~1).
circle_label	the center circle label
pal	color palette
name	label the name
percentage	label the percentage
text_params	parameters parse to geom_text

circle_label_params parameters parse to [geom_text](#)
 bar_params parameters parse to [geom_rect](#)

Value

a ggplot

Examples

```
if (interactive()) {
  data.frame(
    a = c("a", "a", "b", "b", "c"), b = c("a", LETTERS[2:5]), c = rep("a", 5),
    number = 1:5
  ) %>% gghuan2()
}
```

 ggmosaic

ggmosaic for mosaic plot

Description

ggmosaic for mosaic plot

Usage

```
ggmosaic(
  tab,
  rect_params = list(),
  rect_space = 0,
  show_number = c("number", "percentage", "none")[1],
  number_params = list(),
  x_label = c("top", "bottom", "none")[1],
  y_label = c("right", "left", "none")[1],
  label_params = list(),
  chisq_test = TRUE
)
```

Arguments

tab your dataframe, must have 3 columns, the third column must be numeric
 rect_params parameters parse to [geom_rect](#)
 rect_space rect_space, default 0.
 show_number show "number" or "percentage" or "none"
 number_params parameters parse to [geom_text](#)
 x_label show x label on "top" or "bottom" or "none"

y_label show y label on "right" or "left" or "none"
label_params parameters parse to [geom_text](#)
chisq_test whether show chisq test

Value

a ggplot

Examples

```
data(mtcars)  
tab <- dplyr::count(mtcars, gear, cyl)  
ggmosaic(tab,  
  show_number = "number", x_label = "top",  
  y_label = "right", chisq_test = TRUE  
)
```

ggplot_lim *Get a ggplot xlim and ylim*

Description

Get a ggplot xlim and ylim

Usage

```
ggplot_lim(p)
```

Arguments

p ggplot

Value

list

ggplot_translator *Translate axis label of a ggplot*

Description

Translate axis label of a ggplot

Usage

```
ggplot_translator(  
  gg,  
  which = c("x", "y"),  
  from = "en",  
  to = "zh",  
  keep_original_label = FALSE,  
  original_sep = "\n",  
  verbose = TRUE  
)
```

Arguments

gg	a ggplot object to be translated
which	vector contains one or more of 'x', 'y', 'label', 'fill', 'color'..., or 'facet_x', 'facet_y', 'labs' and 'all' to select which texts to be translated.
from	source language
to	target language
keep_original_label	keep the source language labels
original_sep	default, '\n'
verbose	verbose

Value

ggplot

Examples

```
## Not run:  
df <- data.frame(  
  Subject = c("English", "Math"),  
  Score = c(59, 98), Motion = c("sad", "happy")  
)  
ggp <- ggplot(df, mapping = aes(x = Subject, y = Score, label = Motion)) +  
  geom_text() +  
  geom_point() +  
  labs(x = "Subject", y = "Score", title = "Final Examination")
```

```
ggplot_translator(ggp, which = "all")  
## End(Not run)
```

grepl.data.frame	<i>Grepl applied on a data.frame</i>
------------------	--------------------------------------

Description

Grepl applied on a data.frame

Usage

```
grepl.data.frame(pattern, x, ...)
```

Arguments

pattern	search pattern
x	your data.frame
...	additional arguments for gerpl()

Value

a logical matrix

Examples

```
matrix(letters[1:6], 2, 3) |> as.data.frame() -> a  
grepl.data.frame("c", a)  
grepl.data.frame("\\w", a)
```

group_box	<i>Plot a boxplot</i>
-----------	-----------------------

Description

Plot a boxplot

Usage

```

group_box(
  tab,
  group = NULL,
  metadata = NULL,
  mode = 1,
  group_order = NULL,
  facet_order = NULL,
  paired = FALSE,
  paired_line_param = list(),
  alpha = FALSE,
  method = "wilcox",
  alpha_param = list(),
  point_param = NULL,
  p_value1 = FALSE,
  p_value2 = FALSE,
  only_sig = TRUE,
  stat_compare_means_param = NULL,
  trend_line = FALSE,
  trend_line_param = list()
)

```

Arguments

tab	your dataframe
group	which colname choose for group or a vector
metadata	the dataframe contains the group
mode	1~9, plot style, try yourself
group_order	the order of x group
facet_order	the order of the facet
paired	if paired is TRUE, points in different groups will be connected by lines. So the row names order is important.
paired_line_param	parameters parse to geom_line .
alpha	whether plot a group alphabeta by test of method
method	test method:wilcox, tukeyHSD, LSD, (default: wilcox), see multitest
alpha_param	parameters parse to geom_text
point_param	parameters parse to geom_point ,
p_value1	multi-test of all group
p_value2	two-test of each pair
only_sig	only_sig for p_value2
stat_compare_means_param	parameters parse to stat_compare_means
trend_line	add a trend line
trend_line_param	parameters parse to geom_smooth

Value

a ggplot

Examples

```
a <- data.frame(a = 1:18, b = runif(18, 0, 5))
group_box(a, group = rep(c("a", "b", "c"), each = 6))
```

group_test	<i>Performs multiple mean comparisons for a data.frame</i>
------------	--

Description

Performs multiple mean comparisons for a data.frame

Usage

```
group_test(
  df,
  group,
  metadata = NULL,
  method = "wilcox.test",
  pattern = NULL,
  p.adjust.method = "none",
  threads = 1,
  verbose = TRUE
)
```

Arguments

df	a data.frame
group	The compare group (categories) in your data, one column name of metadata when metadata exist or a vector whose length equal to columns number of df.
metadata	sample information dataframe contains group
method	the type of test. Default is wilcox.test. Allowed values include: <ul style="list-style-type: none"> • t.test (parametric) and wilcox.test (non-parametric). Perform comparison between two groups of samples. If the grouping variable contains more than two levels, then a pairwise comparison is performed. • anova (parametric) and kruskal.test (non-parametric). Perform one-way ANOVA test comparing multiple groups. • chisq.test, performs chi-squared contingency table tests and goodness-of-fit tests. • 'pearson', 'kendall', or 'spearman' (correlation), see cor.
pattern	a named vector matching the group, e.g. c('G1'=1,'G2'=3,'G3'=2), use the correlation analysis with specific pattern to calculate p-value.

p.adjust.method p.adjust.method, see [p.adjust](#), default BH.
 threads default 1
 verbose logical

Value

data.frame

Examples

```
data(otutab)
group_test(otutab, metadata$Group, method = "kruskal.test")
group_test(otutab[, 1:12], metadata$Group[1:12], method = "wilcox.test")
```

gsub.data.frame *Gsub applied on a data.frame*

Description

Gsub applied on a data.frame

Usage

```
gsub.data.frame(pattern, replacement, x, ...)
```

Arguments

pattern search pattern
 replacement a replacement for matched pattern
 x your data.frame
 ... additional arguments for `gerpl()`

Value

a data.frame

Examples

```
matrix(letters[1:6], 2, 3) |> as.data.frame() -> a
gsub.data.frame("c", "a", a)
```

 guolv

Filter your data

Description

Filter your data

Usage

```
guolv(tab, sum = 10, exist = 1)
```

Arguments

tab	dataframe
sum	the rowsum should bigger than sum (default:10)
exist	the exist number bigger than exist (default:1)

Value

input object

Examples

```
data(otutab)
guolv(otutab)
```

hebing

Group your data

Description

Group your data

Usage

```
hebing(otutab, group, margin = 2, act = "mean", metadata = NULL)
```

Arguments

otutab	data.frame
group	group vector or one of colnames(metadata)
margin	1 for row and 2 for column(default: 2)
act	do (default: mean)
metadata	metadata

Value

data.frame

Examples

```
data(otutab)
hebing(otutab, metadata$Group)
hebing(otutab, "Group", metadata = metadata, act = "sum")
```

hebing2

Group your data

Description

Group your data

Usage

```
hebing2(otutab, group_df, margin = 2, act = "mean")
```

Arguments

otutab	data.frame
group_df	group data.frame with two columns (id and group). The same ID can be mapped to multiple groups.
margin	1 for row and 2 for column(default: 2)
act	do (default: mean)

Value

data.frame

Examples

```
data(otutab)
hebing2(otutab, data.frame(id = c("NS1", "NS2", "NS1", "NS3"), group = c("A", "A", "B", "B")))
```

how_to_set_font_for_plot

How to set font for ggplot

Description

How to set font for ggplot

Usage

how_to_set_font_for_plot()

Value

No return value

how_to_set_options

How to set options in a package

Description

How to set options in a package

Usage

how_to_set_options(package = "My_package")

Arguments

package package name

Value

No return value

how_to_update_parameters

How to update parameters

Description

How to update parameters

Usage

```
how_to_update_parameters()
```

Value

No return value

how_to_use_parallel *How to use parallel*

Description

How to use parallel

Usage

```
how_to_use_parallel(  
  loop = function(i) {  
    return(mean(rnorm(100)))  
  }  
)
```

Arguments

loop the main function

Value

No return value

how_to_use_sbatch	<i>How to use sbatch</i>
-------------------	--------------------------

Description

How to use sbatch

Usage

```
how_to_use_sbatch(mode = 1)
```

Arguments

mode	1~3
------	-----

Value

No return value

igraph_translator	<i>Translate text of igraph</i>
-------------------	---------------------------------

Description

Translate text of igraph

Usage

```
igraph_translator(  
  ig,  
  from = "en",  
  to = "zh",  
  which = c("vertex", "edge", "all")[1],  
  verbose = TRUE  
)
```

Arguments

ig	igraph object to be translated
from	source language
to	target language
which	vertex, edge, or all
verbose	verbose

Value

igraph object

Examples

```
## Not run:
library(igraph)
ig <- make_graph(c("happy", "sad", "sad", "angry", "sad", "worried"))
plot(ig)
ig2 <- igraph_translator(ig)
font_file <- "/System/Library/Fonts/Supplemental/Songti.ttc"
sysfonts::font_add("Songti", font_file)
plot(ig2, vertex.label.family = "Songti")

## End(Not run)
```

is.ggplot.color *Judge if a characteristic is Rcolor*

Description

Judge if a characteristic is Rcolor

Usage

```
is.ggplot.color(color)
```

Arguments

color characteristic

Value

TRUE or FALSE

Examples

```
is.ggplot.color("red")
is.ggplot.color("notcolor")
is.ggplot.color(NA)
is.ggplot.color("#000")
```

legend_size	<i>Scale a legend size</i>
-------------	----------------------------

Description

Scale a legend size

Usage

```
legend_size(scale = 1)
```

Arguments

scale	default: 1.
-------	-------------

Value

"theme" "gg"

lib_ps	<i>Attach packages or install packages have not benn installed</i>
--------	--

Description

Attach packages or install packages have not benn installed

Usage

```
lib_ps(p_list, ..., all_yes = FALSE, library = TRUE)
```

Arguments

p_list	a vector of packages list
...	packages
all_yes	all install try set to yes?
library	should library the package or just get Namespace ?

Value

No return value

list_to_dataframe	<i>Trans list (with NULL) to data.frame</i>
-------------------	---

Description

Trans list (with NULL) to data.frame

Usage

```
list_to_dataframe(lst)
```

Arguments

lst	list (with NULL)
-----	------------------

Value

a data.frame

little_guodong	<i>My cat</i>
----------------	---------------

Description

my little cat named Guo Dong which drawn by my girlfriend.

Format

rastergrob object.

lm_coefficients	<i>Get coefficients of linear regression model</i>
-----------------	--

Description

This function fits a linear regression model using the given data and formula, and returns the coefficients.

Usage

```
lm_coefficients(data, formula, standardize = FALSE, each = TRUE)
```

Arguments

data	A data frame containing the response variable and predictors.
formula	A formula specifying the structure of the linear regression model.
standardize	Whether to standardize the data before fitting the model.
each	each variable do a lm or whole multi-lm

Value

coefficients The coefficients of the linear regression model.

Examples

```
data <- data.frame(
  response = c(2, 4, 6, 7, 9),
  x1 = c(1, 2, 3, 4, 5),
  x2 = c(2, 3, 6, 8, 9),
  x3 = c(3, 6, 5, 12, 12)
)
coefficients_df <- lm_coefficients(data, response ~ x1 + x2 + x3)
print(coefficients_df)
plot(coefficients_df)
```

make_gitbook

Make a Gitbook using bookdown

Description

Make a Gitbook using bookdown

Usage

```
make_gitbook(
  book_n,
  root_dir = "~/Documents/R/",
  mode = c("gitbook", "bs4")[1],
  author = "Asa12138",
  bib = "~/Documents/R/pc_blog/content/bib/My Library.bib",
  csl = "~/Documents/R/pc_blog/content/bib/science.csl"
)
```

Arguments

book_n	project name
root_dir	root directory
mode	"gitbook","bs4"
author	author
bib	cite papers bib, from Zotero
csl	cite papers format, default science.csl

Value

No return value

make_project	<i>Make a R-analysis project</i>
--------------	----------------------------------

Description

Make a R-analysis project

Usage

```
make_project(pro_n, root_dir = "~/Documents/R/")
```

Arguments

pro_n	project name
root_dir	root directory

Value

No return value

make_py_pkg	<i>Make a new python package</i>
-------------	----------------------------------

Description

Make a new python package

Usage

```
make_py_pkg(  
    pkg_name,  
    path = ".",  
    author = "Your Name",  
    email = "your.email@example.com",  
    description = "A brief description of your library",  
    license = "MIT"  
)
```

Arguments

pkg_name	package name
path	project path, default "."
author	author
email	email
description	description
license	license

Value

No return value

Examples

```
if (interactive()) {
  make_py_pkg("my_python_package",
    path = "~/projects",
    author = "John Doe", description = "My Python library",
    license = "MIT"
  )
}
```

match_df	<i>Match otutab and metadata</i>
----------	----------------------------------

Description

Match otutab and metadata

Usage

```
match_df(otutab, metadata)
```

Arguments

otutab	otutab, rownames are features, colnames are samples
metadata	metadata, rownames are samples

Value

list

Examples

```
data(otutab)
match_df(otutab, metadata)
```

metadata	<i>test data for pcutils package</i>
----------	--------------------------------------

Description

an otutab, metadata and a taxonomy table.

Format

contains an otutab, metadata and a taxonomy table.

otutab contains otutable rawdata

metadata contains metadata

taxonomy contains taxonomy table

mmscale	<i>Min_Max scale</i>
---------	----------------------

Description

Min_Max scale

Usage

```
mmscale(x, min_s = 0, max_s = 1, n = 1, plot = FALSE)
```

Arguments

x	a numeric vector
min_s	scale min
max_s	scale max
n	linear transfer for n=1; the slope will change if n>1 or n<1
plot	whether plot the transfer?

Value

a numeric vector

Examples

```
x <- runif(10)
mmscale(x, 5, 10)
```

multireg	<i>Multiple regression/ variance decomposition analysis</i>
----------	---

Description

Multiple regression/ variance decomposition analysis

Usage

```
multireg(formula, data, TopN = 3)
```

Arguments

formula	formula
data	dataframe
TopN	give top variable importance

Value

ggplot

Examples

```
if (requireNamespace("relaimpo") && requireNamespace("aplot")) {
  data(otutab)
  multireg(env1 ~ Group * ., data = metadata[, 2:7])
}
```

multitest	<i>Multi-groups test</i>
-----------	--------------------------

Description

anova (parametric) and kruskal.test (non-parametric). Perform one-way ANOVA test comparing multiple groups. LSD and TukeyHSD are post hoc test of anova. dunn and nemenyi are post hoc test of kruskal.test. t.test or wilcox is just perform t.test or wilcox.test in each two group (no p.adjust).

Usage

```
multitest(var, group, print = TRUE, return = FALSE)
```

Arguments

var	numeric vector
group	more than two-levels group vector
print	whether print the result
return	return which method result (tukeyHSD or LSD or wilcox?)

Value

No value or a dataframe.

Examples

```
if (requireNamespace("multcompView")) {  
  multitest(runif(30), rep(c("A", "B", "C"), each = 10), return = "wilcox")  
}
```

my_cat

Show my little cat named Guo Dong which drawn by my girlfriend.

Description

Show my little cat named Guo Dong which drawn by my girlfriend.

Usage

```
my_cat(mode = 1, picture = 1)
```

Arguments

mode	1~2
picture	1~2

Value

a ggplot

my_circle_packing	<i>My Circle packing plot</i>
-------------------	-------------------------------

Description

My Circle packing plot

Usage

```
my_circle_packing(
  test,
  anno = NULL,
  mode = 1,
  Group = "level",
  Score = "weight",
  label = "label",
  show_level_name = "all",
  show_tip_label = TRUE,
  str_width = 10
)
```

Arguments

test	a dataframe with hierarchical structure
anno	annotation table with rowname for color or fill.
mode	1~2
Group	fill for mode2
Score	color for mode1
label	the labels column
show_level_name	show which level name? a vector contains some column names.
show_tip_label	show_tip_label, logical
str_width	str_width

Value

ggplot

Examples

```
data(otutab)
cbind(taxonomy, weight = rowSums(otutab))[1:10, ] -> test
if (requireNamespace("igraph") && requireNamespace("ggraph")) {
  my_circle_packing(test)
}
```

`my_circo`*My circo plot*

Description

My circo plot

Usage

```
my_circo(  
  df,  
  reorder = TRUE,  
  pal = NULL,  
  mode = c("circlize", "chorddiag")[1],  
  legend = TRUE,  
  ...  
)
```

Arguments

<code>df</code>	dataframe with three column
<code>reorder</code>	reorder by number?
<code>pal</code>	a vector of colors, you can get from here too: <code>RColorBrewer::brewer.pal(5, "Set2")</code> or <code>ggsci::pal_aaas()(5)</code>
<code>mode</code>	"circlize", "chorddiag"
<code>legend</code>	plot legend?
<code>...</code>	chordDiagram

Value

`chordDiagram`

Examples

```
if (requireNamespace("circlize")) {  
  data.frame(  
    a = c("a", "a", "b", "b", "c"),  
    b = c("a", LETTERS[2:5]), c = 1:5  
  ) %>% my_circo(mode = "circlize")  
  data(otutab)  
  cbind(taxonomy, num = rowSums(otutab))[1:10, c(2, 6, 8)] -> test  
  my_circo(test)  
}
```

`my_lm`*Fit a linear model and plot*

Description

Fit a linear model and plot

Usage

```
my_lm(  
  tab,  
  var,  
  metadata = NULL,  
  smooth_param = list(),  
  facet = TRUE,  
  formula_size = 2.5,  
  ...  
)
```

Arguments

<code>tab</code>	your dataframe
<code>var</code>	which colname choose for var or a vector
<code>metadata</code>	the dataframe contains the var
<code>smooth_param</code>	parameters parse to geom_smooth
<code>facet</code>	whether facet?
<code>formula_size</code>	formula font size, default is 2.5
<code>...</code>	parameters parse to geom_point

Value

a ggplot

Examples

```
if (requireNamespace("ggpmisc")) {  
  my_lm(runif(50), var = 1:50)  
  my_lm(c(1:50) + runif(50, 0, 5), var = 1:50)  
}
```

my_sunburst	<i>My Sunburst plot</i>
-------------	-------------------------

Description

My Sunburst plot

Usage

```
my_sunburst(test, ...)
```

Arguments

test	a dataframe with hierarchical structure
...	look for parameters in plot_ly

Value

htmlwidget

Examples

```
data(otutab)
cbind(taxonomy, num = rowSums(otutab))[1:10, ] -> test
if (requireNamespace("plotly")) {
  my_sunburst(test)
}
```

my_treemap	<i>My Treemap plot</i>
------------	------------------------

Description

My Treemap plot

Usage

```
my_treemap(test, ...)
```

Arguments

test	a three-columns dataframe with hierarchical structure
...	look for parameters in plot_ly

Value

htmlwidget

Examples

```
data(otutab)
cbind(taxonomy, num = rowSums(otutab))[1:10, c(4, 7, 8)] -> test
if (requireNamespace("treemap")) {
  my_treemap(test)
}
```

my_voronoi_treemap *My Voronoi treemap plot*

Description

My Voronoi treemap plot

Usage

```
my_voronoi_treemap(test, ...)
```

Arguments

test a three-columns dataframe with hierarchical structure
... look for parameters in [vt_d3](#)

Value

htmlwidget

Examples

```
data(otutab)
cbind(taxonomy, num = rowSums(otutab))[1:10, c(4, 7, 8)] -> test
if (requireNamespace("voronoiTreemap")) {
  my_voronoi_treemap(test)
}
```

otutab	<i>test data for pcutils package</i>
--------	--------------------------------------

Description

an otutab, metadata and a taxonomy table.

Format

contains an otutab, metadata and a taxonomy table.

otutab contains otutable rawdata

metadata contains metadata

taxonomy contains taxonomy table

plot.coefficients	<i>Plot coefficients as a bar chart or lollipop chart</i>
-------------------	---

Description

This function takes the coefficients and generates a plot to visualize their magnitudes.

Usage

```
## S3 method for class 'coefficients'
plot(x, mode = 1, number = FALSE, x_order = NULL, ...)
```

Arguments

x	The coefficients to be plotted.
mode	The mode of the plot: 1 for bar chart, 2 for lollipop chart.
number	show number
x_order	order of variables
...	add

Value

ggplot

See Also

[lm_coefficients](#)

plotgif *Plot a gif*

Description

Plot a gif

Usage

```
plotgif(plist, file, speed = 1, ...)
```

Arguments

plist	plot list
file	prefix of your .gif file
speed	1
...	add

Value

No return value

plotpdf *Plot a multi-pages pdf*

Description

Plot a multi-pages pdf

Usage

```
plotpdf(  
  plist,  
  file,  
  width = 8,  
  height = 7,  
  browser = "/Applications/Microsoft Edge.app/Contents/MacOS/Microsoft Edge",  
  ...  
)
```

Arguments

plist	plot list
file	prefix of your .pdf file
width	width
height	height
browser	the path of Google Chrome, Microsoft Edge or Chromium in your computer.
...	additional arguments

Value

No return value

prepare_package	<i>Prepare a package</i>
-----------------	--------------------------

Description

Prepare a package

Usage

```
prepare_package(
  pkg_dir = ".",
  exclude = "print.R",
  indent_by = 2,
  check = TRUE,
  ...
)
```

Arguments

pkg_dir	default: "."
exclude	vector for excluding .R files
indent_by	indent_by, default: 2
check	check or not, default: TRUE
...	other parameters for devtools::check

Value

No value

pre_number_str	<i>Prepare a numeric string</i>
----------------	---------------------------------

Description

Prepare a numeric string

Usage

```
pre_number_str(str, split_str = ",", continuous_str = "-")
```

Arguments

str	a string contain ',' and '-'
split_str	split_str ","
continuous_str	continuous_str "-"

Value

vector

Examples

```
pre_number_str("a1,a3,a5,a6-a10")
```

read.file	<i>Read some special format file</i>
-----------	--------------------------------------

Description

Read some special format file

Usage

```
read.file(  
  file,  
  format = NULL,  
  just_print = FALSE,  
  all_yes = FALSE,  
  density = 120,  
  ...  
)
```

Arguments

file	file path
format	"blast", "diamond", "fa", "fasta", "fna", "faa", "bib", "gff", "gtf", "jpg", "png", "pdf", "svg"...
just_print	just print the file
all_yes	all_yes?
density	the resolution for reading pdf or svg
...	additional arguments

Value

data.frame

read_fasta	<i>Read fasta file</i>
------------	------------------------

Description

Read fasta file

Usage

```
read_fasta(fasta_file)
```

Arguments

fasta_file	file path
------------	-----------

Value

data.frame

remove.outliers	<i>Remove outliers</i>
-----------------	------------------------

Description

Remove outliers

Usage

```
remove.outliers(x, factor = 1.5)
```

Arguments

x a numeric vector
factor default 1.5

Value

a numeric vector

Examples

```
remove.outliers(c(1, 10:15))
```

rgb2code *Transform a rgb vector to a Rcolor code*

Description

Transform a rgb vector to a Rcolor code

Usage

```
rgb2code(x, rev = FALSE)
```

Arguments

x vector or three columns data.frame
rev reverse, transform a Rcolor code to a rgb vector

Value

Rcolor code like "#69C404"

Examples

```
rgb2code(c(12, 23, 34))  
rgb2code("#69C404", rev = TRUE)
```

rm_low	<i>Remove the low relative items in each column</i>
--------	---

Description

Remove the low relative items in each column

Usage

```
rm_low(otutab, relative_threshold = 0.0001)
```

Arguments

otutab	otutab
relative_threshold	threshold, default: 1e-4

Value

data.frame

Examples

```
data(otutab)
rm_low(otutab)
```

sample_map	<i>Plot the sampling map</i>
------------	------------------------------

Description

Plot the sampling map

Usage

```
sample_map(
  metadata,
  mode = 1,
  map_params = list(),
  group = NULL,
  point_params = list(),
  label = NULL,
  label_params = list(),
  leaflet_pal = NULL,
  shp_file = NULL,
  crs = 4326,
```

```

xlim = NULL,
ylim = NULL,
add_scale = TRUE,
scale_params = list(),
add_north_arrow = TRUE,
north_arrow_params = list()
)

```

Arguments

metadata	metadata must contains "Longitude","Latitude"
mode	1~3. 1 use basic data from ggplot2. 2 use a shp_file. 3 use the leaflet.
map_params	parameters parse to geom_polygon (mode=1) or geom_sf (mode=2)
group	one column name of metadata which mapping to point color
point_params	parameters parse to geom_point
label	one column name of metadata which mapping to point label
label_params	parameters parse to geom_sf_text
leaflet_pal	leaflet color palette
shp_file	a geojson file parse to sf::read_sf
crs	crs coordinate: https://asa-blog.netlify.app/p/r-map/#crs
xlim	xlim
ylim	ylim
add_scale	add annotation_scale
scale_params	parameters parse to ggspatial::annotation_scale
add_north_arrow	add annotation_north_arrow
north_arrow_params	parameters parse to ggspatial::annotation_north_arrow

Value

map

Examples

```

data(otutab)
anno_df <- metadata[, c("Id", "long", "lat", "Group")]
colnames(anno_df) <- c("Id", "Longitude", "Latitude", "Group")
if (requireNamespace("ggspatial")) {
  sample_map(anno_df, mode = 1, group = "Group", xlim = c(90, 135), ylim = c(20, 50))
}

```

`sanxian`*Three-line table*

Description

Three-line table

Usage

```
sanxian(  
  df,  
  digits = 3,  
  nrow = 10,  
  ncol = 10,  
  fig = FALSE,  
  mode = 1,  
  background = "#D7261E",  
  ...  
)
```

Arguments

<code>df</code>	a data.frame
<code>digits</code>	how many digits should remain
<code>nrow</code>	show how many rows
<code>ncol</code>	show how many columns
<code>fig</code>	output as a figure
<code>mode</code>	1~2
<code>background</code>	background color
<code>...</code>	additional arguments e.g.(rows=NULL)

Value

a ggplot

Examples

```
if (require("kableExtra")) {  
  data(otutab)  
  sanxian(otutab)  
}
```

scale_color_pc	<i>Scale a fill color</i>
----------------	---------------------------

Description

Scale a fill color

Usage

```
scale_color_pc(  
  palette = c("col1", "col2", "col3", "bluered"),  
  alpha = 1,  
  n = 11,  
  ...  
)
```

Arguments

palette	col1~3; or a vector of colors, you can get from: <code>RColorBrewer::brewer.pal(5, "Set2")</code> or <code>ggsci::pal_aaas()(5)</code>
alpha	alpha
n	how many colors you need
...	additional

Value

scale_color

scale_fill_pc	<i>Scale a fill color</i>
---------------	---------------------------

Description

Scale a fill color

Usage

```
scale_fill_pc(  
  palette = c("col1", "col2", "col3", "bluered"),  
  alpha = 1,  
  n = 11,  
  ...  
)
```

Arguments

palette	coll~3; or a vector of colors, you can get from: <code>RColorBrewer::brewer.pal(5, "Set2")</code> or <code>ggsci::pal_aaas()(5)</code>
alpha	alpha
n	how many colors you need
...	additional

Value

scale_fill

search_browse	<i>Search and browse the web for specified terms</i>
---------------	--

Description

This function takes a vector of search terms, an optional search engine (default is Google), and an optional base URL to perform web searches. It opens the default web browser with search results for each term.

Usage

```
search_browse(search_terms, engine = "google", base_url = NULL)
```

Arguments

search_terms	A character vector of search terms to be searched.
engine	A character string specifying the search engine to use (default is "google"). Supported engines: "google", "bing".
base_url	A character string specifying the base URL for web searches. If not provided, the function will use a default URL based on the chosen search engine.

Value

No return value

Examples

```
## Not run:
search_terms <- c(
  "s__Pandoraea_pnomenusa",
  "s__Alicyclophilus_sp._B1"
)

# Using Google search engine
search_browse(search_terms, engine = "google")
```

```
# Using Bing search engine
search_browse(search_terms, engine = "bing")

## End(Not run)
```

set_pcutils_config *Set config*

Description

Set config

Usage

```
set_pcutils_config(item, value)
```

Arguments

item	item
value	value

Value

No value

show_pcutils_config *Show config*

Description

Show config

Usage

```
show_pcutils_config()
```

Value

config

split_text	<i>Split text into parts, each not exceeding a specified character count</i>
------------	--

Description

Split text into parts, each not exceeding a specified character count

Usage

```
split_text(text, nchr_each = 200)
```

Arguments

text	Original text
nchr_each	Maximum character count for each part

Value

List of divided parts

Examples

```
original_text <- paste0(sample(c(letters, "\n"), 400, replace = TRUE), collapse = "")
parts <- split_text(original_text, nchr_each = 200)
lapply(parts, nchar)
```

squash	<i>Squash one column in a data.frame using other columns as id.</i>
--------	---

Description

Squash one column in a data.frame using other columns as id.

Usage

```
squash(df, column, split = ",")
```

Arguments

df	data.frame
column	column name, not numeric position
split	split string

Value

data.frame

Examples

```
df <- data.frame(a = c(1:2, 1:2), b = letters[1:4])
squash(df, "b", ",")
```

stackplot

Plot a stack plot

Description

Plot a stack plot

Plot a area plot

Usage

```
stackplot(
  otutab,
  metadata = NULL,
  group = "Group",
  get_data = FALSE,
  bar_params = list(width = 0.7, position = "stack"),
  topN = 8,
  others = TRUE,
  relative = TRUE,
  legend_title = "",
  stack_order = TRUE,
  group_order = FALSE,
  facet_order = FALSE,
  style = c("group", "sample")[1],
  flow = FALSE,
  flow_params = list(lode.guidance = "frontback", color = "darkgray"),
  number = FALSE,
  repel = FALSE,
  format_params = list(digits = 2),
  text_params = list(position = position_stack())
)
```

```
areaplot(
  otutab,
  metadata = NULL,
  group = "Group",
  get_data = FALSE,
  bar_params = list(position = "stack"),
```

```

topN = 8,
others = TRUE,
relative = TRUE,
legend_title = "",
stack_order = TRUE,
group_order = FALSE,
facet_order = FALSE,
style = c("group", "sample")[1],
number = FALSE,
format_params = list(digits = 2),
text_params = list(position = position_stack())
)

```

Arguments

otutab	otutab
metadata	metadata
group	one group name of columns of metadata
get_data	just get the formatted data?
bar_params	parameters parse to geom_bar
topN	plot how many top species
others	should plot others?
relative	transfer to relative or absolute
legend_title	fill legend_title
stack_order	the order of stack fill
group_order	the order of x group, can be T/F, or a vector of x, or a name, or "cluster"
facet_order	the order of the facet
style	"group" or "sample"
flow	should plot a flow plot?
flow_params	parameters parse to geom_flow
number	show the number?
repel	use the <code>ggrepel::geom_text_repel</code> instead of <code>geom_text</code>
format_params	parameters parse to format
text_params	parameters parse to geom_text

Value

a ggplot
a ggplot

Examples

```
data(otutab)
stackplot(otutab, metadata, group = "Group")

if (interactive()) {
  stackplot(otutab, metadata,
    group = "Group", style = "sample",
    group_order = TRUE, flow = TRUE, relative = FALSE
  )
}

data(otutab)
areaplot(otutab, metadata, group = "Id")

areaplot(otutab, metadata,
  group = "Group", style = "sample",
  group_order = TRUE, relative = FALSE
)
```

strsplit2*Split Composite Names*

Description

Split Composite Names

Usage

```
strsplit2(x, split, colnames = NULL, ...)
```

Arguments

x	character vector
split	character to split each element of vector on, see strsplit
colnames	colnames for the result
...	other arguments are passed to strsplit

Value

data.frame

Examples

```
strsplit2(c("a;b", "c;d"), ";", colnames = c("col1", "col2"))
```

t2	<i>Transpose data.frame</i>
----	-----------------------------

Description

Transpose data.frame

Usage

t2(data)

Arguments

data data.frame

Value

data.frame

taxonomy	<i>test data for pcutils package</i>
----------	--------------------------------------

Description

an otutab, metadata and a taxonomy table.

Format

contains an otutab, metadata and a taxonomy table.

otutab contians otutable rawdata

metadata contians metadata

taxonomy contians taxonomy table

tax_pie	<i>Pie plot</i>
---------	-----------------

Description

Pie plot

Usage

```
tax_pie(otutab, topN = 6, ...)
```

Arguments

otutab	otutab
topN	topN
...	add

Value

a ggplot

Examples

```
data(otutab)  
tax_pie(otutab, topN = 7) + scale_fill_pc()
```

tidai	<i>Replace a vector by named vector</i>
-------	---

Description

Replace a vector by named vector

Usage

```
tidai(x, y, fac = FALSE, keep_origin = FALSE)
```

Arguments

x	a vector need to be replaced
y	named vector
fac	consider the factor?
keep_origin	keep_origin?

Value

vector

Examples

```
tidai(c("a", "a", "b", "d"), c("a" = "red", b = "blue"))
tidai(c("a", "a", "b", "c"), c("red", "blue"))
tidai(c("A" = "a", "B" = "b"), c("a" = "red", b = "blue"))
tidai(factor(c("A" = "a", "B" = "b", "C" = "c")), c("a" = "red", b = "blue", c = "green"))
```

trans

Trans format your data

Description

Trans format your data

Usage

```
trans(df, method = "normalize", margin = 2, ...)
```

Arguments

df	dataframe
method	"cpm", "minmax", "acpm", "total", "log", "max", "frequency", "normalize", "range", "rank", "rrank", "standardize", "pa", "chi.square", "hellinger", "log", "clr", "rclr", "alr"
margin	1 for row and 2 for column(default: 2)
...	additional

Value

data.frame

See Also

[decostand](#)

Examples

```
data(otutab)
trans(otutab, method = "cpm")
```

translator	<i>Translator</i>
------------	-------------------

Description

language: en, zh, jp, fra, th..., see <https://www.cnblogs.com/pieguan/p/10338255.html>

Usage

```
translator(words, from = "en", to = "zh", split = TRUE, verbose = TRUE)
```

Arguments

words	words
from	source language, default "en"
to	target language, default "zh"
split	split to blocks when your words are too much
verbose	verbose

Value

vector

Examples

```
## Not run:
translator(c("love", "if"), from = "en", to = "zh")

## End(Not run)
```

trans_format	<i>Transfer the format of file</i>
--------------	------------------------------------

Description

Transfer the format of file

Usage

```
trans_format(
  file,
  to_format,
  format = NULL,
  ...,
  browser = "/Applications/Microsoft Edge.app/Contents/MacOS/Microsoft Edge"
)
```

Arguments

file	input file
to_format	transfer to
format	input file format
...	additional argument
browser	the path of Google Chrome, Microsoft Edge or Chromium in your computer.

Value

file at work directory

twotest	<i>Two-group test</i>
---------	-----------------------

Description

Two-group test

Usage

```
twotest(var, group)
```

Arguments

var	numeric vector
group	two-levels group vector

Value

No return value

Examples

```
twotest(runif(20), rep(c("a", "b"), each = 10))
```

update_NEWS_md	<i>Update the NEWS.md for a package</i>
----------------	---

Description

Update the NEWS.md for a package

Usage

```
update_NEWS_md(
  package_dir = ".",
  new_features = character(),
  bug_fixes = character(),
  other_changes = character(),
  ...
)
```

Arguments

package_dir	default: "."
new_features	new_features
bug_fixes	bug_fixes
other_changes	other_changes
...	additional info

Value

No value

update_param	<i>Update the parameters</i>
--------------	------------------------------

Description

Keep the different parameters while use the same name in update first.

Usage

```
update_param(default, update)
```

Arguments

default	default (data.frame, list, vector)
update	update (data.frame, list, vector)

Value

same class of your input (data.frame, list or vector)

Examples

```
update_param(list(a = 1, b = 2), list(b = 5, c = 5))
```

venn	<i>Plot a general venn (upset, flower)</i>
------	--

Description

Plot a general venn (upset, flower)

Usage

```
venn(...)

## S3 method for class 'list'
venn(aa, mode = "venn", elements_label = TRUE, ...)

## S3 method for class 'data.frame'
venn(otutab, mode = "venn", elements_label = TRUE, ...)
```

Arguments

...	add
aa	list
mode	"venn", "venn2", "euler", "upset", "flower", "network"
elements_label	logical, show elements label in network?
otutab	table

Value

a plot
a plot
a plot

Examples

```
if (interactive()) {  
  aa <- list(a = 1:3, b = 3:7, c = 2:4)  
  venn(aa, mode = "venn")  
  venn(aa, mode = "euler")  
  venn(aa, mode = "network")  
  venn(aa, mode = "upset")  
  data(otutab)  
  venn(otutab, mode = "flower")  
}
```

write_fasta

Write a data.frame to fasta

Description

Write a data.frame to fasta

Usage

```
write_fasta(df, file_path, str_per_line = 70)
```

Arguments

df	data.frame
file_path	output file path
str_per_line	how many base or animo acid in one line, if NULL, one sequence in one line.

Value

No return value

Index

add_alpha, 4
add_analysis, 4
add_theme, 5
anova, 25
areaplot (stackplot), 59

change_fac_lev, 5
check_directory_structure, 6
china_map, 7
chisq.test, 25
chordDiagram, 42
copy_df, 7
copy_vector, 8
cor, 25
count2, 8

dabiao, 9
deconstand, 64
del_ps, 10
df2distance, 10
df2link, 11
distance2df, 11
download2, 12
download_ncbi_genome_file, 13

explode, 14

fittest, 15
format, 60

generate_labels, 15
geom_bar, 60
geom_col, 18
geom_flow, 60
geom_line, 24
geom_point, 24, 43
geom_rect, 18, 20
geom_smooth, 24, 43
geom_text, 7, 18–21, 24, 60
get_cols, 16
get_legend2, 17

gghist, 17
gghistogram, 17
gghuan, 18
gghuan2, 19
ggmosaic, 20
ggplot_lim, 21
ggplot_translator, 22
grepl.data.frame, 23
group_box, 23
group_test, 25
gsub.data.frame, 26
guolv, 27

hebing, 27
hebing2, 28
how_to_set_font_for_plot, 29
how_to_set_options, 29
how_to_update_parameters, 30
how_to_use_parallel, 30
how_to_use_sbatch, 31

igraph_translator, 31
is.ggplot.color, 32

kruskal.test, 25

legend_size, 33
lib_ps, 33
list_to_dataframe, 34
little_guodong, 34
lm_coefficients, 34, 46

make_gitbook, 35
make_project, 36
make_py_pkg, 36
match_df, 37
metadata, 38
mmscale, 38
multireg, 39
multitest, 24, 39
my_cat, 40

my_circle_packing, 41
my_circo, 42
my_lm, 43
my_sunburst, 44
my_treemap, 44
my_voronoi_treemap, 45

otutab, 46

p.adjust, 26
plot.coefficients, 46
plot_ly, 44
plotgif, 47
plotpdf, 47
pre_number_str, 49
prepare_package, 48

read.file, 49
read_fasta, 50
remove.outliers, 50
rgb2code, 51
rm_low, 52

sample_map, 52
sanxian, 54
scale_color_pc, 55
scale_fill_pc, 55
search_browse, 56
set_pcutils_config, 57
show_pcutils_config, 57
split_text, 58
squash, 58
stackplot, 59
stat_compare_means, 24
strsplit, 61
strsplit2, 61

t.test, 25
t2, 62
tax_pie, 63
taxonomy, 62
tidai, 63
trans, 64
trans_format, 65
translator, 65
twotest, 66

update_NEWS_md, 67
update_param, 67

venn, 68
vt_d3, 45

wilcox.test, 25
write_fasta, 69