Package ‘exploratory’

October 10, 2023

Title A Tool for Large-Scale Exploratory Analyses

Version 0.3.31

Description Conduct numerous exploratory analyses in an instant with a point-and-click interface. With one simple command, this tool launches a Shiny App on the local machine. Drag and drop variables in a data set to categorize them as possible independent, dependent, moderating, or mediating variables. Then run dozens (or hundreds) of analyses instantly to uncover any statistically significant relationships among variables. Any relationship thus uncovered should be tested in follow-up studies.
This tool is designed only to facilitate exploratory analyses and should NEVER be used for p-hacking. Many of the functions used in this package are previous versions of functions in the R Packages 'kim' and 'ezr'.
Selected References:
Chang et al. (2021) <`https://CRAN.R-project.org/package=shiny`>.
Dowle et al. (2021) <`https://CRAN.R-project.org/package=data.table`>.
Kim (2023) <`https://jinkim.science/docs/kim.pdf`>.
Kim (2021) <doi:10.5281/zenodo.4619237>.
Kim (2020) <`https://CRAN.R-project.org/package=ezr`>.
Simmons et al. (2011) <doi:10.1177/0956797611417632>
Tingley et al. (2019) <`https://CRAN.R-project.org/package=mediation`>.
Wickham et al. (2020) <`https://CRAN.R-project.org/package=ggplot2`>.

License GPL-3

URL `https://exploratoryonly.com`

BugReports `https://github.com/jinkim3/exploratory/issues`

Imports data.table, DT, ggplot2, ggridges, lemon, lm.beta, mediation, remotes, shiny, shinydashboard, weights

Suggests moments

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Jin Kim [aut, cre] (<https://orcid.org/0000-0002-5013-3958>)

Maintainer Jin Kim <jin.m.kim@yale.edu>

Repository CRAN

Date/Publication 2023-10-10 10:30:05 UTC

R topics documented:

`cohen_d_from_cohen_textbook`
Cohen's d from Jacob Cohen's textbook (1988)

Description

Calculates Cohen's d as described in Jacob Cohen's textbook (1988), Statistical Power Analysis for the Behavioral Sciences, 2nd Edition Cohen, J. (1988) doi:10.4324/9780203771587

Usage

```
cohen_d_from_cohen_textbook(
  sample_1 = NULL,
  sample_2 = NULL,
  data = NULL,
  iv_name = NULL,
  dv_name = NULL
)
```

Arguments

`sample_1`	a vector of values in the first of two samples
`sample_2`	a vector of values in the second of two samples
`data`	a data object (a data frame or a data.table)
`iv_name`	name of the independent variable
`dv_name`	name of the dependent variable

Value

the output will be a Cohen's d value (a numeric vector of length one)

Examples

```
cohen_d_from_cohen_textbook(1:10, 3:12)
cohen_d_from_cohen_textbook(
  data = mtcars, iv_name = "vs", dv_name = "mpg"
)
```

`compare_groups` *Compare groups*

Description

Compares groups by (1) creating histogram by group; (2) summarizing descriptive statistics by group; and (3) conducting pairwise comparisons (t-tests and Mann-Whitney tests).

Usage

```
compare_groups(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  sigfigs = 3,
  mann_whitney = TRUE,
  t_test_stats = FALSE
)
```

Arguments

`data` a data object (a data frame or a data.table)

`iv_name` name of the independent variable (grouping variable)

`dv_name` name of the dependent variable (measure variable of interest)

`sigfigs` number of significant digits to round to

`mann_whitney` if `mann_whitney` = `TRUE`, Mann-Whitney test results will be included in the pairwise comparison data.table. If `mann_whitney` = `FALSE`, Mann-Whitney tests will not be performed.

`t_test_stats` if `t_test_stats` = `TRUE`, t-test statistic and degrees of freedom will be included in the pairwise comparison data.table.

Value

the output will be a list of (1) ggplot object (histogram by group) (2) a data.table with descriptive statistics by group; and (3) a data.table with pairwise comparison results

Examples

```
compare_groups(data = iris, iv_name = "Species", dv_name = "Sepal.Length")
```

desc_stats *Descriptive statistics*

Description

Returns descriptive statistics for a numeric vector.

Usage

```
desc_stats(
  vector = NULL,
  output_type = "vector",
  sigfigs = 3,
  ci = TRUE,
  pi = TRUE,
  notify_na_count = NULL,
  print_dt = TRUE
)
```

Arguments

`vector` a numeric vector

`output_type` if `output_type = "vector"`, return a vector of descriptive statistics; if `output_type = "dt"`, return a data.table of descriptive statistics (default = "vector")

`sigfigs` number of significant digits to round to (default = 3) `ci` logical. Should 95% CI be included in the descriptive stats? (default = TRUE)

`pi` logical. Should 95% PI be included in the descriptive stats? (default = TRUE)

`notify_na_count`
if `TRUE`, notify how many observations were removed due to missing values. By default, NA count will be printed only if there are any NA values.

`print_dt` if `TRUE`, print the descriptive stats data.table

Value

if `output_type = "vector"`, the output will be a named numeric vector of descriptive statistics; if `output_type = "dt"`, the output will be data.table of descriptive statistics.

Examples

```
desc_stats(1:100)
desc_stats(1:100, ci = TRUE, pi = TRUE, sigfigs = 2)
desc_stats(c(1:100, NA))
desc_stats(vector = c(1:100, NA), output_type = "dt")
```

`desc_stats_by_group` *Descriptive statistics by group*

Description

Returns descriptive statistics by group

Usage

```
desc_stats_by_group(
  data = NULL,
  var_for_stats = NULL,
  grouping_vars = NULL,
  sigfigs = NULL,
  cols_to_round = NULL
)
```

Arguments

`data` a data object (a data frame or a data.table)

`var_for_stats` name of the variable for which descriptive statistics will be calculated

`grouping_vars` name(s) of grouping variables

`sigfigs` number of significant digits to round to

`cols_to_round` names of columns whose values will be rounded

Value

the output will be a data.table showing descriptive statistics of the variable for each of the groups formed by the grouping variables.

Examples

```
desc_stats_by_group(data = mtcars, var_for_stats = "mpg",
grouping_vars = c("vs", "am"))
```

exploratory *Launch the exploratory analysis tool*

Description

Launches the exploratory analysis tool in a browser on the local machine

Usage

```
exploratory(
  data = datasets::mtcars,
  sigfig = 3,
  select_list_max = 1e+05,
  saved_analyses_file_name = "exploratory_analyses_saved.csv",
  run_analysis_file_name = "exploratory_analyses_run.csv"
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`sigfig`	number of significant digits to round to
`select_list_max`	maximum number of variable names to display for dropdown menus
`saved_analyses_file_name`	name of the .csv file in which saved analyses will be recorded (default = "exploratory_analyses_saved.csv")
`run_analysis_file_name`	name of the .csv file in which all conducted analyses will be recorded (default = "exploratory_analyses_run.csv")

Value

There will be no output from this function. Rather, the exploratory analysis tool (a Shiny App) will open in a browser on the local machine.

Examples

```
if (interactive()) {exploratory(data = mtcars)}
```

`histogram` *Histogram*

Description

Create a histogram

Usage

```
histogram(
  vector = NULL,
  number_of_bins = 30,
  x_tick_marks = NULL,
  y_tick_marks = NULL,
  fill_color = "cyan4",
  border_color = "black",
  y_axis_title_vjust = 0.85,
  x_axis_title = NULL,
  y_axis_title = NULL,
  cap_axis_lines = FALSE,
  notify_na_count = NULL
)
```

Arguments

`vector` a numeric vector

`number_of_bins` number of bins for the histogram (default = 30)

`x_tick_marks` a vector of values at which to place tick marks on the x axis (e.g., setting `x_tick_marks = seq(0, 10, 5)` will put tick marks at 0, 5, and 10.)

`y_tick_marks` a vector of values at which to place tick marks on the y axis (e.g., setting `y_tick_marks = seq(0, 10, 5)` will put tick marks at 0, 5, and 10.)

`fill_color` color for inside of the bins (default = "cyan4")

`border_color` color for borders of the bins (default = "black")

`y_axis_title_vjust`
position of the y axis title (default = 0.85).

`x_axis_title` title for x axis (default = "Value")

`y_axis_title` title for y axis (default = "Count")

`cap_axis_lines` logical. Should the axis lines be capped at the outer tick marks? (default = FALSE)

`notify_na_count`
if `TRUE`, notify how many observations were removed due to missing values. By default, NA count will be printed only if there are any NA values.

Value

the output will be a histogram, a ggplot object.

Examples

```
histogram(1:100)
histogram(c(1:100, NA))
histogram(vector = mtcars[["mpg"]])
histogram(vector = mtcars[["mpg"]], x_tick_marks = seq(10, 36, 2))
histogram(vector = mtcars[["mpg"]], x_tick_marks = seq(10, 36, 2),
y_tick_marks = seq(0, 8, 2), y_axis_title_vjust = 0.5,
y_axis_title = "Freq", x_axis_title = "Values of mpg")
```

histogram_by_group *Histogram by group*

Description

Creates histograms by group to compare distributions

Usage

```
histogram_by_group(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  order_of_groups_top_to_bot = NULL,
  number_of_bins = 40,
  space_between_histograms = 0.15,
  draw_baseline = FALSE
)
```

Arguments

`data` a data object (a data frame or a data.table)

`iv_name` name of the independent variable

`dv_name` name of the dependent variable

`order_of_groups_top_to_bot` a character vector indicating the desired presentation order of levels in the independent variable (from the top to bottom). Omitting a group in this argument will remove the group in the set of histograms.

`number_of_bins` number of bins for the histograms (default = 40)

`space_between_histograms` space between histograms (minimum = 0, maximum = 1, default = 0.15)

`draw_baseline` logical. Should the baseline and the trailing lines to either side of the histogram be drawn? (default = FALSE)

Value

the output will be a set of vertically arranged histograms (a ggplot object), i.e., one histogram for each level of the independent variable.

Examples

```
histogram_by_group(data = mtcars, iv_name = "cyl", dv_name = "mpg")
histogram_by_group(
  data = mtcars, iv_name = "cyl", dv_name = "mpg",
  order_of_groups_top_to_bot = c("8", "4"), number_of_bins = 10,
  space_between_histograms = 0.5
)
histogram_by_group(
data = iris, iv_name = "Species", dv_name = "Sepal.Length")
```

`id_across_datasets` *ID across datasets*

Description

Create an ID column in each of the data sets. The ID values will span across the data sets.

Usage

```
id_across_datasets(
  dt_list = NULL,
  id_col_name = "id",
  id_col_position = "first",
  silent = FALSE
)
```

Arguments

`dt_list`	a list of data.table objects
`id_col_name`	name of the column that will contain ID values
`id_col_position`	position of the newly created ID column. If `id_col_position = "first"`, the new ID column will be placed as the first column in respective data sets. If `id_col_position = "last"`, the new ID column will be placed as the last column in respective data sets.
`silent`	If `silent = TRUE`, a summary of starting and ending ID values in each data set will not be printed. If `silent = FALSE`, a summary of starting and ending ID values in each data set will be printed. (default = FALSE)

Value

the output will be a list of data.table objects.

Examples

```
# running the examples below requires importing the data.table package.
prep(data.table)
id_across_datasets(
dt_list = list(setDT(copy(mtcars)), setDT(copy(iris))))
id_across_datasets(
dt_list = list(setDT(copy(mtcars)), setDT(copy(iris)), setDT(copy(women))),
id_col_name = "newly_created_id_col",
id_col_position = "last")
```

kurtosis *Kurtosis*

Description

Calculate kurtosis of the sample using a formula for either the (1) biased estimator or (2) an unbiased estimator of the population kurtosis. Formulas were taken from DeCarlo (1997), doi:10.1037/1082-989X.2.3.292

Usage

```
kurtosis(vector = NULL, unbiased = TRUE)
```

Arguments

`vector`	a numeric vector
`unbiased`	logical. If `unbiased = TRUE`, the unbiased estimate of the population kurtosis will be calculated. If `unbiased = FALSE`, the biased estimate of the population kurtosis will be calculated. By default, `unbiase = TRUE`.

Value

a numeric value, i.e., kurtosis of the given vector

Examples

```
# calculate the unbiased estimator (e.g., kurtosis value that
# Excel 2016 will produce)
exploratory::kurtosis(c(1, 2, 3, 4, 5, 10))
# calculate the biased estimator (e.g., kurtosis value that
# R Package 'moments' will produce)
exploratory::kurtosis(c(1, 2, 3, 4, 5, 10), unbiased = FALSE)
# compare with kurtosis from 'moments' package
moments::kurtosis(c(1, 2, 3, 4, 5, 10))
```

`mann_whitney` *Mann-Whitney U Test (Also called Wilcoxon Rank-Sum Test)*

Description

A nonparametric equivalent of the independent t-test

Usage

```
mann_whitney(data = NULL, iv_name = NULL, dv_name = NULL, sigfigs = 3)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name`	name of the independent variable (grouping variable)
`dv_name`	name of the dependent variable (measure variable of interest)
`sigfigs`	number of significant digits to round to

Value

the output will be a data.table object with all pairwise Mann-Whitney test results

Examples

```
mann_whitney(data = iris, iv_name = "Species", dv_name = "Sepal.Length")
```

`mediation_analysis` *Mediation analysis*

Description

Conducts a mediation analysis to estimate an independent variable's indirect effect on dependent variable through a mediator variable. The current version of the package only supports a simple mediation model consisting of one independent variable, one mediator variable, and one dependent variable. Uses the source code from 'mediation' package v4.5.0, Tingley et al. (2019) https://cran.r-project.org/package=mediation

Usage

```
mediation_analysis(
  data = NULL,
  iv_name = NULL,
  mediator_name = NULL,
  dv_name = NULL,
  covariates_names = NULL,
  robust_se = TRUE,
  iterations = 1000,
  sigfigs = 3,
  output_type = "summary_dt",
  silent = FALSE
)
```

Arguments

data a data object (a data frame or a data.table)

iv_name name of the independent variable

mediator_name name of the mediator variable

dv_name name of the dependent variable

covariates_names
names of covariates to control for

robust_se if TRUE, heteroskedasticity-consistent standard errors will be used in quasi-Bayesian simulations. By default, it will be set as FALSE if nonparametric bootstrap is used and as TRUE if quasi-Bayesian approximation is used.

iterations number of bootstrap samples. The default is set at 1000, but consider increasing the number of samples to 5000, 10000, or an even larger number, if slower handling time is not an issue.

sigfigs number of significant digits to round to

output_type if output_type = "summary_dt", return the summary data.table; if output_type = "mediate_output", return the output from the mediate function in the 'mediate' package; if output_type = "indirect_effect_p", return the p value associated with the indirect effect estimated in the mediation model (default = "summary_dt")

silent if silent = FALSE, mediation analysis summary, estimation method, sample size, and number of simulations will be printed; if silent = TRUE, nothing will be printed. (default = FALSE)

Value

if output_type = "summary_dt", which is the default, the output will be a data.table showing a summary of mediation analysis results; if output_type = "mediate_output", the output will be the output from the mediate function in the 'mediate' package; if output_type = "indirect_effect_p", the output will be the p-value associated with the indirect effect estimated in the mediation model (a numeric vector of length one).

Examples

```
mediation_analysis(
  data = mtcars, iv_name = "cyl",
  mediator_name = "disp", dv_name = "mpg", iterations = 100
)
mediation_analysis(
  data = iris, iv_name = "Sepal.Length",
  mediator_name = "Sepal.Width", dv_name = "Petal.Length",
  iterations = 100
)
```

---

merge_data_tables *Merge data tables*

Description

Merge two data.table objects. If there are any duplicated ID values and column names across the two data tables, the cell values in the first data.table will remain intact and the cell values in the second data.table will be discarded for the resulting merged data table.

Usage

```
merge_data_tables(dt1 = NULL, dt2 = NULL, id = NULL, silent = TRUE)
```

Arguments

dt1	the first data.table which will remain intact
dt2	the second data.table which will be joined outside of (around) the first data.table. If there are any duplicated ID values and column names across the two data tables, the cell values in the first data.table will remain intact and the cell values in the second data.table will be discarded for the resulting merged data table.
id	name of the column that will contain the ID values in the two data tables. The name of the ID column must be identical in the two data tables.
silent	If silent = TRUE, no message will be printed regarding how many ID values and column names were duplicated. If silent = FALSE, messages will be printed regarding how many ID values and column names were duplicated. (default = FALSE)

Value

a data.table object, which merges (joins) the second data.table around the first data.table.

Examples

```
data_1 <- data.table::data.table(
id_col = c(4, 2, 1, 3),
a = 3:6,
b = 5:8,
c = c("w", "x", "y", "z"))
data_2 <- data.table::data.table(
id_col = c(1, 4, 99),
d = 6:8,
b = c("p", "q", "r"),
e = c(TRUE, FALSE, FALSE))
merge_data_tables(dt1 = data_1, dt2 = data_2, id = "id_col")
```

---

merge_data_table_list *Merge a list of data tables*

Description

Successively merge a list of data.table objects in a recursive fashion. That is, merge the (second data table in the list) around the first data table in the list; then, around this resulting data table, merge the third data table in the list; and so on.

Usage

```
merge_data_table_list(dt_list = NULL, id = NULL, silent = TRUE)
```

Arguments

dt_list	a list of data.table objects
id	name of the column that will contain the ID values in the data tables. The name of the ID column must be identical in the all data tables.
silent	If silent = TRUE, no message will be printed regarding how many ID values and column names were duplicated. If silent = FALSE, messages will be printed regarding how many ID values and column names were duplicated. (default = FALSE)

Details

If there are any duplicated ID values and column names across the data tables, the cell values in the earlier data table will remain intact and the cell values in the later data table will be discarded for the resulting merged data table in each recursion.

Value

a data.table object, which successively merges (joins) a data table around (i.e., outside) the previous data table in the list of data tables.

Examples

```
data_1 <- data.table::data.table(
id_col = c(4, 2, 1, 3),
a = 3:6,
b = 5:8,
c = c("w", "x", "y", "z"))
data_2 <- data.table::data.table(
id_col = c(1, 4, 99),
d = 6:8,
b = c("p", "q", "r"),
e = c(TRUE, FALSE, FALSE))
data_3 <- data.table::data.table(
id_col = c(200, 3),
f = 11:12,
b = c(300, "abc"))
merge_data_table_list(
dt_list = list(data_1, data_2, data_3), id = "id_col")
```

multiple_regression *Summarize multiple regression results in a data.table*

Description

Summarize multiple regression results in a data.table

Usage

```
multiple_regression(
  data = NULL,
  formula = NULL,
  sigfigs = NULL,
  round_digits_after_decimal = NULL
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`formula`	a formula object for the regression equation
`sigfigs`	number of significant digits to round to
`round_digits_after_decimal`	round to nth digit after decimal (alternative to `sigfigs`)

Value

the output will be a data.table showing multiple regression results.

Examples

```
multiple_regression(data = mtcars, formula = mpg ~ gear * cyl)
```

order_rows_specifically_in_dt

Order rows specifically in a data table

Description

Order rows in a data.table in a specific order

Usage

```
order_rows_specifically_in_dt(
  dt = NULL,
  col_to_order_by = NULL,
  specific_order = NULL
)
```

Arguments

`dt`	a data.table object
`col_to_order_by`	a character value indicating the name of the column by which to order the data.table
`specific_order`	a vector indicating a specific order of the values in the column by which to order the data.table.

Value

the output will be a data.table object whose rows will be ordered as specified.

Examples

```
order_rows_specifically_in_dt(mtcars, "carb", c(3, 2, 1, 4, 8, 6))
```

prep

Prepare package(s) for use

Description

Installs, loads, and attaches package(s). If package(s) are not installed, installs them prior to loading and attaching.

Usage

```
prep(
  ...,
  pkg_names_as_object = FALSE,
  silent_if_successful = FALSE,
  silent_load_pkgs = NULL
)
```

Arguments

`...` names of packages to load and attach, separated by commas, e.g., `"ggplot2"`, `data.table`. The input can be any number of packages, whose names may or may not be wrapped in quotes.

`pkg_names_as_object`
logical. If `pkg_names_as_object = TRUE`, the input will be evaluated as one object containing package names. If `pkg_names_as_object = FALSE`, the input will be considered as literal packages names (default = FALSE).

`silent_if_successful`
logical. If `silent_if_successful = TRUE`, no message will be printed if preparation of package(s) is successful. If `silent_if_successful = FALSE`, a message indicating which package(s) were successfully loaded and attached will be printed (default = FALSE).

`silent_load_pkgs`
a character vector indicating names of packages to load silently (i.e., suppress messages that get printed when loading the packaged). By default, `silent_load_pkgs = NULL`

Value

there will be no output from this function. Rather, packages given as inputs to the function will be installed, loaded, and attached.

Examples

```
prep(data.table)
prep("data.table", silent_if_successful = TRUE)
prep("base", utils, ggplot2, "data.table")
pkgs <- c("ggplot2", "data.table")
prep(pkgs, pkg_names_as_object = TRUE)
prep("data.table", silent_load_pkgs = "data.table")
```

pretty_round_p_value *Pretty round p-value*

Description

Pretty round p-value

Usage

```
pretty_round_p_value(
  p_value_vector = NULL,
  round_digits_after_decimal = 3,
  include_p_equals = FALSE
)
```

Arguments

`p_value_vector` one number or a numeric vector

`round_digits_after_decimal`
how many digits after the decimal point should the p-value be rounded to?

`include_p_equals`
if `TRUE`, output will be a string of mathematical expression including "p", e.g., "p < .01" (default = FALSE)

Value

the output will be a character vector with p values, e.g., a vector of strings like "< .001" (or "p < .001").

Examples

```
pretty_round_p_value(
  p_value_vector = 0.049,
  round_digits_after_decimal = 2, include_p_equals = FALSE
)
pretty_round_p_value(c(0.0015, 0.0014), include_p_equals = TRUE)
```

read_csv *Read a csv file*

Description

Read a csv file

Usage

```
read_csv(name = NULL, head = FALSE, ...)
```

Arguments

`name`	a character string of the csv file name without the ".csv" extension. For example, if the csv file to read is "myfile.csv", enter `name = "myfile"`
`head`	logical. if `head = TRUE`, prints the first five rows of the data set.
`...`	optional arguments for the `fread` function from the data.table package. Any arguments for data.table's `fread` function can be used, e.g., `fill = TRUE`, `nrows = 100`

Value

the output will be a data.table object, that is, an output from the data.table function, `fread`

Examples

```
## Not run:
mydata <- read_csv("myfile")

## End(Not run)
```

`scatterplot` *Scatterplot*

Description

Creates a scatter plot and calculates a correlation between two variables

Usage

```
scatterplot(
  data = NULL,
  x_var_name = NULL,
  y_var_name = NULL,
  point_label_var_name = NULL,
  weight_var_name = NULL,
  alpha = 1,
  annotate_stats = FALSE,
  annotate_y_pos = 5,
  line_of_fit_type = "lm",
  ci_for_line_of_fit = FALSE,
  x_axis_label = NULL,
  y_axis_label = NULL,
  point_label_size = NULL,
  point_size_range = c(3, 12),
  jitter_x_percent = 0,
  jitter_y_percent = 0,
  cap_axis_lines = FALSE
)
```

Arguments

`data` a data object (a data frame or a data.table)

`x_var_name` name of the variable that will go on the x axis

`y_var_name` name of the variable that will go on the y axis

`point_label_var_name` name of the variable that will be used to label individual observations

`weight_var_name` name of the variable by which to weight the individual observations for calculating correlation and plotting the line of fit

`alpha` opacity of the dots (0 = completely transparent, 1 = completely opaque)

`annotate_stats` if `TRUE`, the correlation and p-value will be annotated at the top of the plot

`annotate_y_pos` position of the annotated stats, expressed as a percentage of the range of y values by which the annotated stats will be placed above the maximum value of y in the data set (default = 5). If `annotate_y_pos = 5`, and the minimum and maximum y values in the data set are 0 and 100, respectively, the annotated stats will be placed at 5% of the y range (100 - 0) above the maximum y value, y = 0.05 * (100 - 0) + 100 = 105.

`line_of_fit_type` if `line_of_fit_type = "lm"`, a regression line will be fit; if `line_of_fit_type = "loess"`, a local regression line will be fit; if `line_of_fit_type = "none"`, no line will be fit

`ci_for_line_of_fit` if `ci_for_line_of_fit = TRUE`, confidence interval for the line of fit will be shaded

`x_axis_label` alternative label for the x axis

`y_axis_label` alternative label for the y axis

`point_label_size` size for dots' labels on the plot. If no input is entered for this argument, it will be set as `point_label_size = 5` by default. If the plot is to be weighted by some variable, this argument will be ignored, and dot sizes will be determined by the argument `point_size_range`

`point_size_range` minimum and maximum size for dots on the plot when they are weighted

`jitter_x_percent` horizontally jitter dots by a percentage of the range of x values

`jitter_y_percent` vertically jitter dots by a percentage of the range of y values

`cap_axis_lines` logical. Should the axis lines be capped at the outer tick marks? (default = TRUE)

Value

the output will be a scatter plot, a ggplot object.

Examples

```
scatterplot(data = mtcars, x_var_name = "wt", y_var_name = "mpg")
scatterplot(
  data = mtcars, x_var_name = "wt", y_var_name = "mpg",
  point_label_var_name = "hp", weight_var_name = "drat",
  annotate_stats = TRUE
)
scatterplot(
  data = mtcars, x_var_name = "wt", y_var_name = "mpg",
  point_label_var_name = "hp", weight_var_name = "cyl",
  point_label_size = 7, annotate_stats = TRUE
)
```

`se_of_mean` *Standard error of the mean*

Description

Standard error of the mean

Usage

```
se_of_mean(vector, na.rm = TRUE, notify_na_count = NULL)
```

Arguments

`vector` a numeric vector

`na.rm` if `TRUE`, NA values will be removed before calculation

`notify_na_count` if `TRUE`, notify how many observations were removed due to missing values. By default, NA count will be printed only if there are any NA values.

Value

the output will be a numeric vector of length one, which will be the standard error of the mean for the given numeric vector.

Examples

```
se_of_mean(c(1:10, NA))
```

skewness *Skewness*

Description

Calculate skewness using one of three formulas: (1) the traditional Fisher-Pearson coefficient of skewness; (2) the adjusted Fisher-Pearson standardized moment coefficient; (3) the Pearson 2 skewness coefficient. Formulas were taken from Doane & Seward (2011), doi:10.1080/10691898.2011.11889611

Usage

```
skewness(vector = NULL, type = "adjusted")
```

Arguments

vector	a numeric vector
type	a character string indicating the type of skewness to calculate. If type = "adjusted", the adjusted Fisher-Pearson standardized moment coefficient will be calculated. If type = "traditional", the traditional Fisher-Pearson coefficient of skewness will be calculated. If type = "pearson_2", the Pearson 2 skewness coefficient will be calculated. By default, type = "adjusted".

Value

a numeric value, i.e., skewness of the given vector

Examples

```
# calculate the adjusted Fisher-Pearson standardized moment coefficient
exploratory::skewness(c(1, 2, 3, 4, 5, 10))
# calculate the traditional Fisher-Pearson coefficient of skewness
exploratory::skewness(c(1, 2, 3, 4, 5, 10), type = "traditional")
# compare with skewness from 'moments' package
moments::skewness(c(1, 2, 3, 4, 5, 10))
# calculate the Pearson 2 skewness coefficient
exploratory::skewness(c(1, 2, 3, 4, 5, 10), type = "pearson_2")
```

tabulate_vector *Tabulate vector*

Description

Shows frequency and proportion of unique values in a table format

Usage

```
tabulate_vector(
  vector = NULL,
  na.rm = TRUE,
  sort_by_decreasing_count = NULL,
  sort_by_increasing_count = NULL,
  sort_by_decreasing_value = NULL,
  sort_by_increasing_value = NULL,
  total_included = TRUE,
  sigfigs = NULL,
  round_digits_after_decimal = NULL,
  output_type = "dt"
)
```

Arguments

`vector` a character or numeric vector

`na.rm` if `TRUE`, NA values will be removed before calculating frequencies and proportions.

`sort_by_decreasing_count`
if `TRUE`, the output table will be sorted in the order of decreasing frequency.

`sort_by_increasing_count`
if `TRUE`, the output table will be sorted in the order of increasing frequency.

`sort_by_decreasing_value`
if `TRUE`, the output table will be sorted in the order of decreasing value.

`sort_by_increasing_value`
if `TRUE`, the output table will be sorted in the order of increasing value.

`total_included` if `TRUE`, the output table will include a row for total counts.

`sigfigs` number of significant digits to round to

`round_digits_after_decimal`
round to nth digit after decimal (alternative to `sigfigs`)

`output_type` if `output_type = "df"`, return a data.frame. By default, `output_type = "dt"`, which will return a data.table.

Value

if `output_type = "dt"`, which is the default, the output will be a data.table showing the count and proportion (percent) of each element in the given vector; if `output_type = "df"`, the output will be a data.frame showing the count and proportion (percent) of each value in the given vector.

Examples

```
tabulate_vector(c("a", "b", "b", "c", "c", "c", NA))
tabulate_vector(c("a", "b", "b", "c", "c", "c", NA),
  sort_by_increasing_count = TRUE
)
tabulate_vector(c("a", "b", "b", "c", "c", "c", NA),
  sort_by_decreasing_value = TRUE
)
tabulate_vector(c("a", "b", "b", "c", "c", "c", NA),
  sort_by_increasing_value = TRUE
)
tabulate_vector(c("a", "b", "b", "c", "c", "c", NA),
  sigfigs = 4
)
tabulate_vector(c("a", "b", "b", "c", "c", "c", NA),
  round_digits_after_decimal = 1
)
tabulate_vector(c("a", "b", "b", "c", "c", "c", NA),
  output_type = "df"
)
```

theme_kim *Theme Kim*

Description

A custom ggplot theme

Usage

```
theme_kim(
  legend_position = "none",
  base_size = 20,
  axis_tick_font_size = 20,
  axis_title_font_size = 24,
  y_axis_title_vjust = 0.85,
  axis_title_margin_size = 24,
  cap_axis_lines = TRUE
)
```

Arguments

`legend_position`
position of the legend (default = "none")

`base_size` base font size

`axis_tick_font_size`
font size for axis tick marks

`axis_title_font_size`
font size for axis title

`y_axis_title_vjust`
position of the y axis title (default = 0.85). If default is used, `y_axis_title_vjust = 0.85`, the y axis title will be positioned at 85% of the way up from the bottom of the plot.

`axis_title_margin_size`
: size of the margin between axis title and the axis line

`cap_axis_lines`
: logical. Should the axis lines be capped at the outer tick marks? (default = TRUE)

Value

a ggplot object; there will be no meaningful output from this function. Instead, this function should be used with another ggplot object, e.g., `ggplot(mtcars , aes(x = disp, y = mpg)) + theme_kim()`

Examples

```
prep(ggplot2)
ggplot2::ggplot(mtcars, aes(x = cyl, y = mpg)) + geom_point() + theme_kim()
```

`t_test_pairwise` *t test, pairwise*

Description

Conducts a t-test for every possible pairwise comparison with Bonferroni correction

Usage

```
t_test_pairwise(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  sigfigs = 3,
  mann_whitney = TRUE,
  t_test_stats = FALSE,
  t_test_df_decimals = 1,
  sd = FALSE
)
```

Arguments

`data`
: a data object (a data frame or a data.table)

`iv_name`
: name of the independent variable

`dv_name`
: name of the dependent variable

`sigfigs`
: number of significant digits to round to

`mann_whitney`
: if `TRUE`, Mann-Whitney test results will be included in the output data.table. If `TRUE`, Mann-Whitney tests will not be performed.

`t_test_stats` if `t_test_stats` = TRUE, t-test statistic and degrees of freedom will be included in the output data.table.

`t_test_df_decimals` number of decimals for the degrees of freedom in t-tests (default = 1)

`sd` if `sd` = TRUE, standard deviations will be included in the output data.table.

Value

the output will be a data.table showing results of all pairwise comparisons between levels of the independent variable.

Examples

```
t_test_pairwise(data = iris, iv_name = "Species", dv_name = "Sepal.Length")
t_test_pairwise(data = iris, iv_name = "Species",
dv_name = "Sepal.Length", t_test_stats = TRUE, sd = TRUE)
t_test_pairwise(data = iris, iv_name = "Species", dv_name = "Sepal.Length",
mann_whitney = FALSE)
```

`update_exploratory` *Update the package 'exploratory'*

Description

Updates the current package 'exploratory' by installing the most recent version of the package from GitHub This function requires installing Package 'remotes' v2.4.2 (or possibly a higher version) by Csardi et al. (2021), https://cran.r-project.org/package=remotes

Usage

```
update_exploratory(force = TRUE, upgrade_other_pkg = FALSE, confirm = TRUE)
```

Arguments

`force` logical. If `force` = TRUE, force installing the update. If `force` = FALSE, do not force installing the update. By default, `force` = TRUE.

`upgrade_other_pkg` input for the `upgrade` argument to be passed on to `remotes::install_github`. One of "default", "ask", "always", "never", TRUE, or FALSE. "default" respects the value of the R_REMOTES_UPGRADE environment variable if set, and falls back to "ask" if unset. "ask" prompts the user for which out of date packages to upgrade. For non-interactive sessions "ask" is equivalent to "always". TRUE and FALSE correspond to "always" and "never" respectively. By default, `upgrade_other_pkg` = FALSE.

`confirm` logical. If `confirm` = TRUE, the user will need to confirm the update. If `confirm` = FALSE, the confirmation step will be skipped. By default, `confirm` = TRUE.

Value

there will be no output from this function. Rather, executing this function will update the current 'exploratory' package by installing the most recent version of the package from GitHub.

Examples

```
## Not run:
if (interactive()) {update_exploratory()}

## End(Not run)
```

wilcoxon_rank_sum_test

Wilcoxon Rank-Sum Test (Also called the Mann-Whitney U Test)

Description

A nonparametric equivalent of the independent t-test

Usage

```
wilcoxon_rank_sum_test(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  sigfigs = 3
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name`	name of the independent variable (grouping variable)
`dv_name`	name of the dependent variable (measure variable of interest)
`sigfigs`	number of significant digits to round to

Value

the output will be a data.table object with all pairwise Wilcoxon rank-sum test results

Examples

```
wilcoxon_rank_sum_test(
data = iris, iv_name = "Species", dv_name = "Sepal.Length")
```

Index