Package ‘kim’

October 2, 2024

Title A Toolkit for Behavioral Scientists

Version 0.5.431

Description A collection of functions for analyzing data typically collected
or used by behavioral scientists. Examples of the functions include
a function that compares groups in a factorial experimental design,
a function that conducts two-way analysis of variance (ANOVA),
and a function that cleans a data set generated by Qualtrics surveys.
Some of the functions will require installing additional package(s).
Such packages and other references are cited within the section
describing the relevant functions. Many functions in this package
rely heavily on these two popular R packages:
Dowle et al. (2021) <`https://CRAN.R-project.org/package=data.table`>.
Wickham et al. (2021) <`https://CRAN.R-project.org/package=ggplot2`>.

License GPL-3

URL `https://github.com/jinkim3/kim`, `https://jinkim.science`

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

Imports data.table, remotes

Suggests boot, ggplot2, moments, testthat (>= 3.0.0)

Encoding UTF-8

RoxygenNote 7.3.0

Config/testthat/edition 3

NeedsCompilation no

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

Maintainer Jin Kim <`jin.kim1@northeastern.edu`>

Repository CRAN

Date/Publication 2024-10-01 23:50:02 UTC

Contents

`akaike_weights` *Akaike Weights*

Description

Compare adequacy of different models by calculating their Akaike weights and the associated evidence ratio.

Usage

```
akaike_weights(aic_values = NULL, print_output_explanation = TRUE)
```

Arguments

`aic_values`	a vector of AIC values
`print_output_explanation`	logical. Should an explanation about how to read the output be printed? (default = TRUE).

Details

Please refer to Wagenmakers & Farrell (2004), doi:10.3758/BF03206482

Value

the output will be a data.table showing AIC weights, their evidence ratio(s), etc.

Examples

```
# default reference AIC value is the minimum AIC value, e.g., 202 below.
akaike_weights(c(204, 202, 206, 206, 214))
```

assign_fn_parameters_as_vars

Assign function parameters as values

Description

Take a function and assign all the parameters defined within it as values in the specified environment (e.g., global environment)

Usage

```
assign_fn_parameters_as_vars(fun = NULL, envir = NULL)
```

Arguments

`fun`	a function
`envir`	an environment in which to assign the parameters as values (default = `.GlobalEnv`)

Details

This function can be useful when you are testing a function and you need to set all the function's parameters in a single operation.

Examples

```
## Not run:
assign_fn_parameters_as_vars(pm)
assign_fn_parameters_as_vars(mean)
assign_fn_parameters_as_vars(sum)
assign_fn_parameters_as_vars(lm)
assign_fn_parameters_as_vars(floodlight_2_by_continuous)

## End(Not run)
```

barplot_for_counts

Barplot for counts

Description

Barplot for counts

Usage

```
barplot_for_counts(data = NULL, x, y)
```

Arguments

`data`	a data object (a data frame or a data.table)
`x`	name of the variable that will be on the x axis of the barplot
`y`	name of the variable that will be on the y axis of the barplot

Examples

```
barplot_for_counts(x = 1:3, y = 7:9)
barplot_for_counts(data = data.frame(
cyl = names(table(mtcars$cyl)), count = as.vector(table(mtcars$cyl))),
x = "cyl", y = "count")
```

`binomial_test` *Binomial test*

Description

Conduct a binomial test. In other words, test whether an observed proportion of "successes" (e.g., proportion of heads in a series of coin tosses) is greater than the expected proportion (e.g., 0.5). This function uses the 'binom.test' function from the 'stats' package.

Usage

```
binomial_test(
  x = NULL,
  success = NULL,
  failure = NULL,
  p = 0.5,
  alternative = "two.sided",
  ci = 0.95,
  round_percentages = 0
)
```

Arguments

`x`	a vector of values, each of which represents an instance of either a "success" or "failure" (e.g., c("s", "f", "s", "s", "f", "s"))
`success`	which value(s) indicate "successes"?
`failure`	(optional) which value(s) indicate "failures"? If no input is provided for this argument, then all the non-NA values that are not declared to be "successes" will be treated as "failures".
`p`	hypothesized probability of success (default = 0.5)
`alternative`	indicates the alternative hypothesis and must be one of "two.sided", "greater", or "less". You can specify just the initial letter. By default, `alternative = "two.sided"`

`ci`	width of the confidence interval (default = 0.95)
`round_percentages`	number of decimal places to which to round the percentages in the summary table (default = 0)

Examples

```
# sample vector
sample_vector <- c(0, 1, 1, 0, 1, 98, 98, 99, NA)
binomial_test(
x = sample_vector,
success = 1, failure = 0)
binomial_test(
x = sample_vector,
success = 1, failure = 0,
p = 0.1,
alternative = "greater")
binomial_test(
x = sample_vector,
success = c(1, 99), failure = c(0, 98),
p = 0.6,
alternative = "less")
```

bracket *Draw a bracket on a ggplot*

Description

Draw a square bracket with a label on a ggplot

Usage

```
bracket(
  xmin = NULL,
  xmax = NULL,
  ymin = NULL,
  ymax = NULL,
  vertical = NULL,
  horizontal = NULL,
  open = NULL,
  bracket_shape = NULL,
  thickness = 2,
  bracket_color = "black",
  label = NULL,
  label_hjust = NULL,
  label_vjust = NULL,
  label_font_size = 5,
  label_font_face = "bold",
  label_color = "black",
  label_parse = FALSE
)
```

Arguments

`xmin`	xmin
`xmax`	xmax
`ymin`	ymin
`ymax`	ymax
`vertical`	vertical
`horizontal`	horizontal
`open`	open
`bracket_shape`	bracket_shape
`thickness`	thickness
`bracket_color`	bracket_color
`label`	label
`label_hjust`	label_hjust
`label_vjust`	label_vjust
`label_font_size`	label_font_size
`label_font_face`	label_font_face
`label_color`	label_font_face
`label_parse`	label_parse

Value

a ggplot object; there will be no meaningful output from this function. Instead, this function should be used with another ggplot object

Examples

```
library(ggplot2)
ggplot(mtcars, aes(x = cyl, y = mpg)) + geom_point() +
bracket(6.1, 6.2, 17, 22, bracket_shape = "]", label = "abc")
```

capitalize *Capitalize a substring*

Description

Capitalizes the first letter (by default) or a substring of a given character string or each element of the character vector

Usage

```
capitalize(x, start = 1, end = 1)
```

Arguments

x	a character string or a character vector
start	starting position of the susbtring (default = 1)
end	ending position of the susbtring (default = 1)

Value

a character string or a character vector

Examples

```
capitalize("abc")
capitalize(c("abc", "xyx"), start = 2, end = 3)
```

change_var_names *Change variable names in a data set*

Description

Change variable names in a data set

Usage

```
change_var_names(
  data = NULL,
  old_var_names = NULL,
  new_var_names = NULL,
  skip_absent = FALSE,
  print_summary = TRUE,
  output_type = "dt"
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`old_var_names`	a vector of old variable names (i.e., variable names to change)
`new_var_names`	a vector of new variable names
`skip_absent`	If `skip_absent` = `TRUE`, old variable names that do not exist in the data set will be skipped (default = TRUE).
`print_summary`	If `print_summary` = `TRUE`, a summary of old and new variable names will printed. (default = TRUE)
`output_type`	type of the output. If `output_type` = `"dt"`, the function's output will be a data.table with changed names. If `output_type` = `"summary"`, the function's output will be a data.table listing old and new variable names. By default, `output_type` = `"dt"`.

Value

a data.table object with changed variable names

Examples

```
change_var_names(
mtcars, old = c("mpg", "cyl"), new = c("mpg_new", "cyl_new"))
```

`check_modes` *Check modes of objects*

Description

Check modes of objects

Usage

```
check_modes(..., mode_to_confirm = NULL)
```

Arguments

`...`	R objects.
`mode_to_confirm`	The function will test whether each input is of this mode. For example, `check_modes(a, mode_to_confirm = "numeric")`, the function will check whether the object a is numeric.

Examples

```
check_modes(1L, mode_to_confirm = "numeric")
check_modes(
TRUE, FALSE, 1L, 1:3, 1.1, c(1.2, 1.3), "abc", 1 + 2i, intToBits(1L),
mode_to_confirm = "numeric")
```

`check_req_pkg` *Check for required packages*

Description

Check whether required packages are installed.

Usage

```
check_req_pkg(pkg = NULL)
```

Arguments

`pkg`	a character vector containing names of packages to check

Value

there will be no output from this function. Rather, the function will check whether the packages given as inputs are installed.

Examples

```
check_req_pkg("data.table")
check_req_pkg(c("base", "utils", "ggplot2", "data.table"))
```

`chi_squared_test` *Chi-squared test*

Description

Conduct a chi-squared test and produce a contingency table

Usage

```
chi_squared_test(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  round_chi_sq_test_stat = 2,
  round_p = 3,
  sigfigs_proportion = 2,
  correct = TRUE,
  odds_ratio_ci = 0.95,
  round_odds_ratio_ci_limits = 2,
  invert = FALSE,
  notify_na_count = NULL
)
```

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 (must be a binary variable)
`round_chi_sq_test_stat`	number of decimal places to which to round the chi-squared test statistic (default = 2)
`round_p`	number of decimal places to which to round the p-value from the chi-squared test (default = 3)
`sigfigs_proportion`	number of significant digits to round to (for the table of proportions). By default `sigfigs_proportion = 2`
`correct`	logical. Should continuity correction be applied? (default = TRUE)
`odds_ratio_ci`	width of the confidence interval for the odds ratio. Input can be any value less than 1 and greater than or equal to 0. By default, `odds_ratio_ci = 0.95`. If `odds_ratio_ci = TRUE`, the default value of 0.95 will be used. If `odds_ratio_ci = FALSE`, no confidence interval will be estimated for the odds ratio.
`round_odds_ratio_ci_limits`	number of decimal places to which to round the limits of the odds ratio's confidence interval (default = 2)
`invert`	logical. Whether the inverse of the odds ratio (i.e., 1 / odds ratio) should be returned.
`notify_na_count`	if `TRUE`, notify how many rows were removed due to missing values. By default, NA count will be printed only if there are any NA values.

Examples

```
chi_squared_test(data = mtcars, iv_name = "cyl", dv_name = "am")
# if the iv has only two levels, odds ratio will also be calculated
chi_squared_test(data = mtcars, iv_name = "vs", dv_name = "am")
```

`chi_squared_test_pairwise` *Chi-squared test, pairwise*

Description

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

Usage

```
chi_squared_test_pairwise(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  focal_dv_value = NULL,
  contingency_table = TRUE,
  contingency_table_sigfigs = 2,
  percent_and_total = FALSE,
  percentages_only = NULL,
  counts_only = NULL,
  sigfigs = 3,
  chi_sq_test_stats = FALSE,
  correct = TRUE
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name`	name of the independent variable (must be a categorical variable)
`dv_name`	name of the dependent variable (must be a binary variable)
`focal_dv_value`	focal value of the dependent variable whose frequencies will be calculated (i.e., the value of the dependent variable that will be considered a "success" or a result of interest)
`contingency_table`	If `contingency_table = TRUE` or if `contingency_table = "percentages"`, the percentage of each binary value within each group will be printed. If `contingency_table = "counts"`, a table of frequencies will be printed. If `contingency_table = FALSE`, no contingency table will be printed.
`contingency_table_sigfigs`	number of significant digits that the contingency table's percentage values should be rounded to (default = 2)
`percent_and_total`	logical. If `percent_and_total = TRUE`, tabulate percentages of the focal DV value and a total count of the two values in DV. By default `percent_and_total = FALSE`
`percentages_only`	tabulate percentages of the focal DV value only
`counts_only`	tabulate counts of the focal DV value only
`sigfigs`	number of significant digits to round to
`chi_sq_test_stats`	if `chi_sq_test_stats = TRUE`, chi-squared test statistic and degrees of freedom will be included in the pairwise comparison data.table.
`correct`	logical. Should continuity correction be applied? (default = TRUE)

Examples

```
chi_squared_test_pairwise(data = mtcars, iv_name = "vs", dv_name = "am")
chi_squared_test_pairwise(data = mtcars, iv_name = "vs", dv_name = "am",
percentages_only = TRUE)
# using 3 mtcars data sets combined
chi_squared_test_pairwise(
data = rbind(mtcars, rbind(mtcars, mtcars)),
iv_name = "cyl", dv_name = "am")
# include the total counts
chi_squared_test_pairwise(
data = rbind(mtcars, rbind(mtcars, mtcars)),
iv_name = "cyl", dv_name = "am", percent_and_total = TRUE)
# display counts
chi_squared_test_pairwise(
data = rbind(mtcars, rbind(mtcars, mtcars)),
iv_name = "cyl", dv_name = "am", contingency_table = "counts")
```

`ci_of_mean` *Confidence Interval of the Mean of a Vector*

Description

Returns the confidence interval of the mean of a numeric vector.

Usage

```
ci_of_mean(x = NULL, confidence_level = 0.95, notify_na_count = NULL)
```

Arguments

`x`	a numeric vector
`confidence_level`	What is the desired confidence level expressed as a decimal? (default = 0.95)
`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 named numeric vector with the lower and upper limit of the confidence interval.

Examples

```
ci_of_mean(x = 1:100, confidence_level = 0.95)
ci_of_mean(mtcars$mpg)
```

clean_data_from_qualtrics *Clean data from Qualtrics*

Description

Clean a data set downloaded from Qualtrics

Usage

```
clean_data_from_qualtrics(
  data = NULL,
  remove_survey_preview_data = TRUE,
  remove_test_response_data = TRUE,
  default_cols_by_qualtrics = NULL,
  default_cols_by_qualtrics_new = NULL,
  warn_accuracy_loss = FALSE,
  click_data_cols = "rm",
  page_submit_cols = "move_to_right"
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`remove_survey_preview_data`	logical. Whether to remove data from survey preview (default = TRUE)
`remove_test_response_data`	logical. Whether to remove data from test response (default = TRUE)
`default_cols_by_qualtrics`	names of columns that Qualtrics includes in the data set by default (e.g., "StartDate", "Finished"). Accepting the default value `default_cols_by_qualtrics = NULL` will set the names to be those that Qualtrics uses as of Dec 25, 2020.
`default_cols_by_qualtrics_new`	new names for columns that Qualtrics includes in the data set by default (e.g., "StartDate", "Finished"). Accepting the default value `default_cols_by_qualtrics_new = NULL` will set the names to be those that Qualtrics uses as of Dec 25, 2020 converted to snake_case (e.g., "start_date", "finished").
`warn_accuracy_loss`	logical. whether to warn the user if converting character to numeric leads to loss of accuracy. (default = FALSE)
`click_data_cols`	if `click_data_cols = "rm"`, columns containing click data (e.g., "_First Click") will be removed. If `click_data_cols = "move_to_right"`, the columns will be moved to the right (end) of the data set.

`page_submit_cols`	if `page_submit_cols = "rm"`, columns containing page submit data (e.g., "_Page Submit"; "response time" data) will be removed. If `page_submit_cols = "move_to_right"`, the columns will be moved to the right (end) of the data set.

Value

a data.table object

Examples

```
clean_data_from_qualtrics(mtcars)
clean_data_from_qualtrics(mtcars, default_cols_by_qualtrics = "mpg",
default_cols_by_qualtrics_new = "mpg2")
```

`coefficent_of_variation` *Coefficient of variation*

Description

Calculates the (population or sample) coefficient of variation of a given numeric vector

Usage

```
coefficent_of_variation(vector, pop_or_sample = "pop")
```

Arguments

`vector`	a numeric vector
`pop_or_sample`	should coefficient of variation be calculated for a "population" or a "sample"?

Value

a numeric value

Examples

```
coefficent_of_variation(1:4, pop_or_sample = "sample")
coefficent_of_variation(1:4, pop_or_sample = "pop")
```

cohen_d — *Calculate Cohen's d and its confidence interval using the package 'psych'*

Description

To run this function, the following package(s) must be installed: Package 'psych' v2.1.9 (or possibly a higher version) by William Revelle (2021), https://cran.r-project.org/package=psych

Usage

```
cohen_d(
  sample_1 = NULL,
  sample_2 = NULL,
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  ci_range = 0.95,
  output_type = "all"
)
```

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
`ci_range`	range of the confidence interval for Cohen's d (default = 0.95)
`output_type`	If `output_type == "all"` or if `output_type == "d_and_ci"`, the output will be a vector of Cohen's d and its confidence interval. If `output_type == "d"`, the output will be Cohen's d. If `output_type == "ci"`, the output will be a vector of the confidence interval around Cohen's d. By default, `output_type == "all"`.

Examples

```
## Not run:
cohen_d(sample_1 = 1:10, sample_2 = 3:12)
cohen_d(data = mtcars, iv_name = "vs", dv_name = "mpg", ci_range = 0.99)
sample_dt <- data.table::data.table(iris)[Species != "setosa"]
cohen_d(data = sample_dt, iv_name = "Species", dv_name = "Petal.Width")

## End(Not run)
```

cohen_d_borenstein — *Calculate Cohen's d as illustrated by Borenstein et al. (2009, ISBN: 978-0-470-05724-7)*

Description

Calculates Cohen's d, its standard error, and confidence interval, as illustrated in the Borenstein et al. (2009, ISBN: 978-0-470-05724-7).

Usage

```
cohen_d_borenstein(
  sample_1 = NULL,
  sample_2 = NULL,
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  direction = "2_minus_1",
  ci_range = 0.95,
  output_type = "all",
  initial_value = 0
)
```

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
`direction`	If `direction == "2_minus_1"`, Cohen's d will reflect the extent to which the mean of IV level 2 is greater than the mean of IV level 2. If `direction == "1_minus_2"`, Cohen's d will reflect the extent to which the mean of IV level 1 is greater than the mean of IV level 2. By default, `direction == "2_minus_1"`.
`ci_range`	range of the confidence interval for Cohen's d (default = 0.95)
`output_type`	If `output_type == "all"` or if `output_type == "d_var_se_and_ci"`, the output will be a vector of Cohen's d and its variance, SE, and confidence interval. If `output_type == "d_se_and_ci"`, the output will be a vector of Cohen's d and its SE and confidence interval. If `output_type == "d_and_ci"`, the output will be a vector of Cohen's d and its confidence interval. If `output_type == "d"`, the output will be Cohen's d. If `output_type == "ci"`, the output will be a vector of the confidence interval around Cohen's d. If `output_type == "se"`, the output will be the standard error of Cohen's d. By default, `output_type == "all"`.
`initial_value`	initial value of the noncentrality parameter for optimization (default = 0). Adjust this value if confidence interval results look strange.

Examples

```
cohen_d_borenstein(sample_1 = 1:10, sample_2 = 3:12)
cohen_d_borenstein(
data = mtcars, iv_name = "vs", dv_name = "mpg", ci_range = 0.99)
sample_dt <- data.table::data.table(iris)[Species != "setosa"]
cohen_d_borenstein(
data = sample_dt, iv_name = "Species", dv_name = "Petal.Width",
initial_value = 10)
```

cohen_d_for_one_sample

Calculate Cohen's d to accompany a one-sample t-test

Description

To run this function, the following package(s) must be installed: Package 'psych' v2.1.9 (or possibly a higher version) by William Revelle (2021), https://cran.r-project.org/package=psych

Usage

```
cohen_d_for_one_sample(x = NULL, mu = NULL)
```

Arguments

x	a numeric vector containing values whose mean will be calculated
mu	the true mean

Examples

```
cohen_d_for_one_sample(x = 1:10, mu = 3)
cohen_d_for_one_sample(x = c(1:10, NA, NA), mu = 3)
```

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"
)
```

`cohen_d_over_n` *Cohen's d as a function of sample size*

Description

Plot Cohen's d as sample size increases.

Usage

```
cohen_d_over_n(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  save_as_png = FALSE,
  png_name = NULL,
  xlab = NULL,
  ylab = NULL,
  width = 16,
  height = 9
)
```

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)
`save_as_png`	if `save = TRUE`, the plot will be saved as a PNG file.
`png_name`	name of the PNG file to be saved. By default, the name will be "cohen_d_over_n_" followed by a timestamp of the current time. The timestamp will be in the format, jan_01_2021_1300_10_000001, where "jan_01_2021" would indicate January 01, 2021; 1300 would indicate 13:00 (i.e., 1 PM); and 10_000001 would indicate 10.000001 seconds after the hour.
`xlab`	title of the x-axis for the histogram by group. If `xlab = FALSE`, the title will be removed. By default (i.e., if no input is given), `dv_name` will be used as the title.
`ylab`	title of the y-axis for the histogram by group. If `ylab = FALSE`, the title will be removed. By default (i.e., if no input is given), `iv_name` will be used as the title.
`width`	width of the plot to be saved. This argument will be directly entered as the `width` argument for the `ggsave` function within `ggplot2` package (default = 16)
`height`	height of the plot to be saved. This argument will be directly entered as the `height` argument for the `ggsave` function within `ggplot2` package (default = 9)

Value

the output will be a list of (1) ggplot object (histogram by group) and (2) a data.table with Cohen's d by sample size

Examples

```
## Not run:
cohen_d_over_n(data = mtcars, iv_name = "am", dv_name = "mpg")

## End(Not run)
```

`cohen_d_torchiano` *Calculate Cohen's d and its confidence interval using the package 'effsize'*

Description

To run this function, the following package(s) must be installed: Package 'effsize' v0.8.1 (or possibly a higher version) by Marco Torchiano (2020), https://cran.r-project.org/package=effsize

Usage

```
cohen_d_torchiano(
  sample_1 = NULL,
  sample_2 = NULL,
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  ci_range = 0.95
)
```

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
`ci_range`	range of the confidence interval for Cohen's d (default = 0.95)

Examples

```
cohen_d_torchiano(1:10, 3:12)
cohen_d_torchiano(
data = mtcars, iv_name = "vs", dv_name = "mpg", ci_range = 0.99)
```

`cohen_d_to_r` *Convert Cohen's d to r*

Description

Convert d (standardized mean difference or Cohen's d) to r (correlation), as illustrated in Borenstein et al. (2009, p. 48, ISBN: 978-0-470-05724-7)

Usage

```
cohen_d_to_r(d = NULL, n1 = NULL, n2 = NULL, d_var = NULL)
```

Arguments

`d`	Cohen's d (the input can be a vector of values)
`n1`	sample size in the first of two group (the input can be a vector of values)
`n2`	sample size in the second of two group (the input can be a vector of values)
`d_var`	(optional argument) variance of d (the input can be a vector of values). If this argument receives an input, variance of r will be returned as well.

Value

the output will be a vector of correlation values (and variances of r if the argument d_var received an input)

Examples

```
## Not run:
cohen_d_to_r(1)
cohen_d_to_r(d = 1:3)
cohen_d_to_r(d = 1:3, n1 = c(100, 200, 300), n2 = c(50, 250, 900))
cohen_d_to_r(1.1547)
cohen_d_to_r(d = 1.1547, d_var = .0550)
cohen_d_to_r(d = 1:2, d_var = 1:2)

## End(Not run)
```

`combine_data_across_cols` *Combine data across columns*

Description

Combine data across columns. If NA is the only value across all focal columns for given row(s), NA will be returned for those row(s).

Usage

```
combine_data_across_cols(data = NULL, cols = NULL)
```

Arguments

`data`	a data object (a data frame or a data.table)
`cols`	a character vector containing names of columns, across which to combine data

Value

the output will be a numeric or character vector.

Examples

```
dt <- data.frame(v1 = c(1, NA), v2 = c(NA, 2))
dt
combine_data_across_cols(data = dt, cols = c("v1", "v2"))
dt <- data.frame(v1 = c(1, 2, NA), v2 = c(NA, 4, 3))
dt
combine_data_across_cols(data = dt, cols = c("v1", "v2"))
dt <- data.frame(v1 = c(1, NA, NA), v2 = c(NA, 2, NA))
dt
combine_data_across_cols(data = dt, cols = c("v1", "v2"))
```

`comma_sep_string_to_numbers` *Convert a comma-separated string of numbers*

Description

Convert a comma-separated string of numbers

Usage

```
comma_sep_string_to_numbers(string)
```

Arguments

`string`	a character string consisting of numbers separated by commas

Value

a character string

Examples

```
comma_sep_string_to_numbers("1, 2, 3,4,  5  6")
```

`compare_datasets` *Compare data sets*

Description

Compares whether or not data sets are identical

Usage

```
compare_datasets(dataset_1 = NULL, dataset_2 = NULL, dataset_list = NULL)
```

Arguments

`dataset_1`	a data object (a data frame or a data.table)
`dataset_2`	another data object (a data frame or a data.table)
`dataset_list`	list of data objects (data.frame or data.table)

Value

the output will be a data.table showing differences in data sets

Examples

```
# catch differences in class attributes of the data sets
compare_datasets(
dataset_1 = data.frame(a = 1:2, b = 3:4),
dataset_2 = data.table::data.table(a = 1:2, b = 3:4))
# catch differences in number of columns
compare_datasets(
dataset_1 = data.frame(a = 1:2, b = 3:4, c = 5:6),
dataset_2 = data.frame(a = 1:2, b = 3:4))
# catch differences in number of rows
compare_datasets(
dataset_1 = data.frame(a = 1:2, b = 3:4),
dataset_2 = data.frame(a = 1:10, b = 11:20))
# catch differences in column names
compare_datasets(
dataset_1 = data.frame(A = 1:2, B = 3:4),
dataset_2 = data.frame(a = 1:2, b = 3:4))
# catch differences in values within corresponding columns
compare_datasets(
dataset_1 = data.frame(a = 1:2, b = c(3, 400)),
dataset_2 = data.frame(a = 1:2, b = 3:4))
compare_datasets(
dataset_1 = data.frame(a = 1:2, b = 3:4, c = 5:6),
dataset_2 = data.frame(a = 1:2, b = c(3, 4), c = c(5, 6)))
# check if data sets in a list are identical
compare_datasets(
dataset_list = list(
dt1 = data.frame(a = 1:2, b = 3:4, c = 5:6),
dt2 = data.frame(a = 1:2, b = 3:4),
dt3 = data.frame(a = 1:2, b = 3:4, c = 5:6)))
```

`compare_dependent_rs` *Compare dependent correlations*

Description

Compares whether two dependent correlations from the same sample are significantly different each other.

Usage

```
compare_dependent_rs(
  data = NULL,
  var_1_name = NULL,
  var_2_name = NULL,
  var_3_name = NULL,
  one_tailed = FALSE,
  round_r = 3,
  round_p = 3,
  round_t = 2,
  print_summary = TRUE,
  return_dt = FALSE
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`var_1_name`	name of the variable whose correlations with two other variables will be compared.
`var_2_name`	name of the first of the two variables whose correlations with `var_1_name` will be compared.
`var_3_name`	name of the second of the two variables whose correlations with `var_1_name` will be compared.
`one_tailed`	logical. Should the p value based on a one-tailed t-test? (default = FALSE)
`round_r`	number of decimal places to which to round correlation coefficients (default = 2)
`round_p`	number of decimal places to which to round p-values (default = 3)
`round_t`	number of decimal places to which to round the t-statistic (default = 2)
`print_summary`	logical. Should the summary be printed? (default = TRUE)
`return_dt`	logical. Should the function return a summary table as an output, as opposed to returning the output through the "invisible" function? (default = FALSE)

Details

Suppose that Variables A, B, and C are measured from a group of subjects. This function tests whether A is related to B differently than to C. Put differently, this function tests H0: r(A, B) = r(A, C)

For more information on formulas used in this function, please refer to Steiger (1980) doi:10.1037/00332909.87.2.245 and Chen & Popovich (2002) doi:10.4135/9781412983808

Value

the output will be a summary of the test comparing two dependent correlations

Examples

```
compare_dependent_rs(
data = mtcars, var_1_name = "mpg", var_2_name = "hp", var_3_name = "wt")
```

compare_effect_sizes *Compare effect sizes*

Description

Compares effect sizes See p. 156 of Borenstein et al. (2009, ISBN: 978-0-470-05724-7).

Usage

```
compare_effect_sizes(
  effect_sizes = NULL,
  effect_size_variances = NULL,
  round_stats = TRUE,
  round_p = 3,
  round_se = 2,
  round_z = 2,
  pretty_round_p_value = TRUE
)
```

Arguments

`effect_sizes`	a vector of estimated effect sizes
`effect_size_variances`	a vector of variances of the effect sizes
`round_stats`	logical. Should the statistics be rounded? (default = TRUE)
`round_p`	number of decimal places to which to round p-values (default = 3)
`round_se`	number of decimal places to which to round the standard errors of the difference (default = 2)
`round_z`	number of decimal places to which to round the z-statistic (default = 2)
`pretty_round_p_value`	logical. Should the p-values be rounded in a pretty format (i.e., lower threshold: "<.001"). By default, `pretty_round_p_value = TRUE`.

Examples

```
compare_effect_sizes(
effect_sizes = c(0.6111, 0.3241, 0.5),
effect_size_variances = c(.0029, 0.0033, 0.01))
```

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,
  stats = "basic",
  cohen_d = TRUE,
  cohen_d_w_ci = TRUE,
  adjust_p = "holm",
  bonferroni = NULL,
  mann_whitney = TRUE,
  t_test_stats = TRUE,
  t_test_df_decimals = 1,
  round_p = 3,
  save_as_png = FALSE,
  png_name = NULL,
  xlab = NULL,
  ylab = NULL,
  x_limits = NULL,
  x_breaks = NULL,
  x_labels = NULL,
  width = 5000,
  height = 3600,
  units = "px",
  res = 300,
  layout_matrix = NULL,
  col_names_nicer = TRUE,
  convert_dv_to_numeric = TRUE
)
```

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

`stats`	statistics to calculate for each group. If `stats = "basic"`, group size, mean, standard deviation, median, minimum, and maximum will be calculated. If `stats = "all"`, in addition to the aforementioned statistics, standard error, 95% confidence and prediction intervals, skewness, and kurtosis will also be calculated. The `stats` argument can also be a character vector with types of statistics to calculate. For example, entering `stats = c("mean", "median")` will calculate mean and median. By default, `stats = "basic"`
`cohen_d`	if `cohen_d = TRUE`, Cohen's d statistics will be included in the pairwise comparison data.table.
`cohen_d_w_ci`	if `cohen_d_w_ci = TRUE`, Cohen's d with 95% CI will be included in the output data.table.
`adjust_p`	the name of the method to use to adjust p-values. If `adjust_p = "holm"`, the Holm method will be used; if `adjust_p = "bonferroni"`, the Bonferroni method will be used. By default, `adjust_p = "holm"`
`bonferroni`	The use of this argument is deprecated. Use the 'adjust_p' argument instead. If `bonferroni = TRUE`, Bonferroni tests will be conducted for t-tests or Mann-Whitney tests.
`mann_whitney`	if `TRUE`, Mann-Whitney test results will be included in the pairwise comparison data.table. If `FALSE`, Mann-Whitney tests will not be performed.
`t_test_stats`	if `t_test_stats = FALSE`, t-test statistic and degrees of freedom will be excluded in the pairwise comparison data.table.
`t_test_df_decimals`	number of decimals for the degrees of freedom in t-tests (default = 1)
`round_p`	number of decimal places to which to round p-values (default = 3)
`save_as_png`	if `save_as_png = "all"` or if `save_as_png = TRUE`, the histogram by group, descriptive statistics by group, and pairwise comparison results will be saved as a PNG file.
`png_name`	name of the PNG file to be saved. By default, the name will be "compare_groups_results_" followed by a timestamp of the current time. The timestamp will be in the format, jan_01_2021_1300_10_000001, where "jan_01_2021" would indicate January 01, 2021; 1300 would indicate 13:00 (i.e., 1 PM); and 10_000001 would indicate 10.000001 seconds after the hour.
`xlab`	title of the x-axis for the histogram by group. If `xlab = FALSE`, the title will be removed. By default (i.e., if no input is given), `dv_name` will be used as the title.
`ylab`	title of the y-axis for the histogram by group. If `ylab = FALSE`, the title will be removed. By default (i.e., if no input is given), `iv_name` will be used as the title.
`x_limits`	a numeric vector with values of the endpoints of the x axis.
`x_breaks`	a numeric vector indicating the points at which to place tick marks on the x axis.
`x_labels`	a vector containing labels for the place tick marks on the x axis.
`width`	width of the PNG file (default = 5000)
`height`	height of the PNG file (default = 3600)
`units`	the units for the `width` and `height` arguments. Can be `"px"` (pixels), `"in"` (inches), `"cm"`, or `"mm"`. By default, `units = "px"`.

`res`	The nominal resolution in ppi which will be recorded in the png file, if a positive integer. Used for units other than the default. By default, `res = 300`
`layout_matrix`	The layout argument for arranging plots and tables using the `grid.arrange` function.
`col_names_nicer`	if `col_names_nicer = TRUE`, column names will be converted from snake_case to an easier-to-eye format.
`convert_dv_to_numeric`	logical. Should the values in the dependent variable be converted to numeric for plotting the histograms? (default = TRUE)
`holm`	if `holm = TRUE`, the relevant p values will be adjusted using Holm method (also known as the Holm-Bonferroni or Bonferroni-Holm method)

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. If `save_as_png = TRUE`, the plot and tables will be also saved on local drive as a PNG file.

Examples

```
## Not run:
compare_groups(data = iris, iv_name = "Species", dv_name = "Sepal.Length")
compare_groups(data = iris, iv_name = "Species", dv_name = "Sepal.Length",
x_breaks = 4:8)

## End(Not run)
```

`compare_independent_rs` *Compare independent correlations*

Description

Compares whether two correlations from two independent samples are significantly different each other. See Field et al. (2012, ISBN: 978-1-4462-0045-2).

Usage

```
compare_independent_rs(
  r1 = NULL,
  n1 = NULL,
  r2 = NULL,
  n2 = NULL,
  one_tailed = FALSE,
  round_p = 3,
  round_z_diff = 2,
  round_r = 2,
  print_summary = TRUE,
  output_type = NULL
)
```

Arguments

r1	correlation in the first sample
n1	size of the first sample
r2	correlation in the second sample
n2	size of the first sample
one_tailed	logical. Should the p value based on a one-tailed t-test? (default = FALSE)
round_p	(only for displaying purposes) number of decimal places to which to round the p-value (default = 3)
round_z_diff	(only for displaying purposes) number of decimal places to which to round the z-score (default = 2)
round_r	(only for displaying purposes) number of decimal places to which to round correlation coefficients (default = 2)
print_summary	logical. Should the summary be printed? (default = TRUE)
output_type	type of the output. If `output_type = "z"`, the function's output will be the z-score of the difference between the two correlations. If `output_type = "p"`, the function's output will be the p-value associated with the z-score of the difference between the two correlations. By default, `output_type = NULL`, and the function will not return any value other than the printed summary.

Value

the output will be the results of a test comparing two independent correlations.

Examples

```
compare_independent_rs(r1 = .1, n1 = 100, r2 = .2, n2 = 200)
compare_independent_rs(
r1 = .1, n1 = 100, r2 = .2, n2 = 200, one_tailed = TRUE)
compare_independent_rs(r1 = .506, n1 = 52, r2 = .381, n2 = 51)
```

---

contingency_table *Contingency table*

Description

Create a contingency table that takes two variables as inputs

Usage

```
contingency_table(
  data = NULL,
  row_var_name = NULL,
  col_var_name = NULL,
  row = NULL,
  col = NULL,
  output_type = "table"
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`row_var_name`	name of the variable whose values will fill the rows of the contingency table
`col_var_name`	name of the variable whose values will fill the columns of the contingency table
`row`	a vector whose values will fill the rows of the contingency table
`col`	a vector whose values will fill the columns of the contingency table
`output_type`	If `output_type` == `"dt"` the output will be a contingency table as a data.table object. If `output_type` == `"table"` the output will be a contingency table as a table object. If `output_type` == `"df"` the output will be a contingency table as a data.frame object. By default, `output_type` == `"table"`.

Examples

```
contingency_table(
data = mtcars,
row_var_name = "am",
col_var_name = "cyl")
contingency_table(row = mtcars$cyl, col = mtcars$am)
contingency_table(mtcars, "am", "cyl", output_type = "dt")
```

---

convert_cols_to_numeric
Convert columns to numeric

Description

Check whether each column in a data.table can be converted to numeric, and if so, convert every such column.

Usage

```
convert_cols_to_numeric(
  data = NULL,
  classes = "character",
  warn_accuracy_loss = TRUE,
  print_summary = TRUE,
  silent = FALSE
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`classes`	a character vector specifying classes of columns that will be converted. For example, if `classes = "character"`, all columns of the class "character" will be converted–if they can be converted. The current version of the function only supports converting character columns to numeric.
`warn_accuracy_loss`	logical. whether to warn the user if converting character to numeric leads to loss of accuracy. (default = TRUE)
`print_summary`	If `print_summary = TRUE`, a summary of converted columns will printed. (default = TRUE)
`silent`	If `silent = FALSE`, a message regarding conversion for a data.frame will be printed. If `silent = TRUE`, this message will be suppressed. By default, `silent = FALSE`.

Examples

```
data_frame_1 <- data.frame(a = c("1", "2"), b = c("1", "b"), c = 1:2)
convert_cols_to_numeric(data = data_frame_1)
data_table_1 <- data.table::data.table(
a = c("1", "2"), b = c("1", "b"), c = 1:2)
convert_cols_to_numeric(data = data_table_1)
```

---

convert_to_excel_formula *Convert character to Excel formula*

Description

Convert elements of a character vector to Excel formulas to preserve the character (string) format when opened in an Excel file.

Usage

```
convert_to_excel_formula(vector = NULL)
```

Arguments

`vector`	a character vector

Value

the output will be a character vector formatted as an Excel formula. For example, if an element in the input vector was `".500"`, this element will be converted to `=".500"`, which will show up as ".500" in Excel, rather than as "0.5"

Examples

```
## Not run:
# compare the two csv files below
# example 1
dt <- data.table::data.table(a = ".500")
data.table::fwrite(dt, "example1.csv") # the csv will show "0.5"
# example 2
dt <- data.table::data.table(a = convert_to_excel_formula(".500"))
data.table::fwrite(dt, "example2.csv") # the csv will show ".500"

## End(Not run)
```

`correlation_kim` *Estimate the correlation between two variables*

Description

Estimate the correlation between two variables

Usage

```
correlation_kim(
  x = NULL,
  y = NULL,
  data = NULL,
  x_var_name = NULL,
  y_var_name = NULL,
  ci_range = 0.95,
  round_r = 2,
  round_p = 3,
  output_type = "summary"
)
```

Arguments

`x`	a numeric vector of data values
`y`	a numeric vector of data values
`data`	(optional) a data object (a data frame or a data.table)
`x_var_name`	(optional) name of the first variable (if using a data set as an input)
`y_var_name`	(optional) name of the second variable (if using a data set as an input)

`ci_range`	range of the confidence interval for the correlation coefficient. If `ci_range = FALSE`, no confidence interval will be estimated. By default, `ci_range = 0.95`.
`round_r`	number of decimal places to which to round correlation coefficients (default = 2)
`round_p`	number of decimal places to which to round p-values (default = 3)
`output_type`	type of the output. If `output_type = "dt"`, the function's output will be a data.table with the results from the correlation analysis. If `output_type = "summary"`, the function's output will be a statement (a string) summarizing the results from the correlation analysis. By default, `output_type = "summary"`

Examples

```
## Not run:
correlation_kim(x = 1:4, y = c(1, 3, 2, 4))
correlation_kim(x = 1:4, y = c(1, 3, 2, 4), ci_range = FALSE)
# output as a data table
correlation_kim(x = 1:4, y = c(1, 3, 2, 4), output_type = "dt")

## End(Not run)
```

`correlation_matrix` *correlation matrix*

Description

Creates a correlation matrix

Usage

```
correlation_matrix(
  data = NULL,
  var_names = NULL,
  row_var_names = NULL,
  col_var_names = NULL,
  round_r = 2,
  round_p = 3,
  output_type = "rp",
  numbered_cols = NULL
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`var_names`	names of the variables for which to calculate all pairwise correlations
`row_var_names`	names of the variables that will go on the rows of the correlation matrix
`col_var_names`	names of the variables that will go on the columns of the correlation matrix

`round_r`	number of decimal places to which to round correlation coefficients (default = 2)
`round_p`	number of decimal places to which to round p-values (default = 3)
`output_type`	which value should be filled in cells of the correlation matrix? If `output_type` = `"r"`, correlation coefficients; if `output_type` = `"p"`, p-values; if `output_type` = `"rp"`, correlation coefficients with significance symbols based on p-values; if `output_type` = `"n"`, sizes of the samples used to calculate the correlation coefficients. By default, `output_type` = `"rp"`
`numbered_cols`	logical. If `numbered_cols` == `TRUE` and if `identical(row_var_names, col_var_names)` == `TRUE`, then the columns will be numbered instead of containing variable names.

Value

the output will be a correlation matrix in a data.table format

Examples

```
correlation_matrix(data = mtcars, var_names = c("mpg", "cyl", "wt"))
correlation_matrix(data = mtcars,
row_var_names = c("mpg", "cyl", "hp"), col_var_names = c("wt", "am"))
correlation_matrix(
data = mtcars, var_names = c("mpg", "cyl", "wt"),
numbered_cols = FALSE)
correlation_matrix(
data = mtcars, var_names = c("mpg", "cyl", "wt"), output_type = "r")
```

`cum_percent_plot` *Cumulative percentage plot*

Description

Plots or tabulates cumulative percentages associated with elements in a vector

Usage

```
cum_percent_plot(vector, output_type = "plot")
```

Arguments

`vector`	a numeric vector
`output_type`	if `output_type` = `"plot"`, return a cumulative percentage plot; if `output_type` = `"dt"`, return a data.table with cumulative percentages. By default, `output_type` = `"plot"`

Examples

```
cum_percent_plot(c(1:100, NA, NA))
cum_percent_plot(mtcars$mpg)
cum_percent_plot(vector= mtcars$mpg, output_type = "dt")
```

`desc_stats` *Descriptive statistics*

Description

Returns descriptive statistics for a numeric vector.

Usage

```
desc_stats(
  vector = NULL,
  output_type = "vector",
  sigfigs = 3,
  se_of_mean = FALSE,
  ci = FALSE,
  pi = FALSE,
  skewness = FALSE,
  kurtosis = FALSE,
  notify_na_count = NULL,
  print_dt = FALSE
)
```

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)
`se_of_mean`	logical. Should the standard errors around the mean be included in the descriptive stats? (default = FALSE)
`ci`	logical. Should 95% CI be included in the descriptive stats? (default = FALSE)
`pi`	logical. Should 95% PI be included in the descriptive stats? (default = FALSE)
`skewness`	logical. Should the skewness statistic be included in the descriptive stats? (default = FALSE)
`kurtosis`	logical. Should the kurtosis statistic be included in the descriptive stats? (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.
`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(1:100, se_of_mean = TRUE,
ci = TRUE, pi = TRUE, sigfigs = 2,
skewness = TRUE, kurtosis = TRUE)
desc_stats(c(1:100, NA))
example_dt <- desc_stats(vector = c(1:100, NA), output_type = "dt")
example_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,
  stats = "all",
  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
`stats`	statistics to calculate. If `stats = "basic"`, group size, mean, standard deviation, median, minimum, and maximum will be calculated. If `stats = "all"`, in addition to the aforementioned statistics, standard error, 95% confidence and prediction intervals, skewness, and kurtosis will also be calculated. The `stats` argument can also be a character vector with types of statistics to calculate. For example, entering `stats = c("mean", "median")` will calculate mean and median. By default, `stats = "all"`
`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"))
desc_stats_by_group(data = mtcars, var_for_stats = "mpg",
grouping_vars = c("vs", "am"), sigfigs = 3)
desc_stats_by_group(data = mtcars, var_for_stats = "mpg",
grouping_vars = c("vs", "am"), stats = "basic", sigfigs = 2)
desc_stats_by_group(data = mtcars, var_for_stats = "mpg",
grouping_vars = c("vs", "am"), stats = "basic", sigfigs = 2,
cols_to_round = "all")
desc_stats_by_group(data = mtcars, var_for_stats = "mpg",
grouping_vars = c("vs", "am"), stats = c("mean", "median"), sigfigs = 2,
cols_to_round = "all")
```

---

detach_user_installed_pkgs

Detach all user-installed packages

Description

Detach all user-installed packages

Usage

```
detach_user_installed_pkgs(exceptions = NULL, force = FALSE, keep_kim = TRUE)
```

Arguments

`exceptions`	a character vector of names of packages to keep attached
`force`	logical. Should a package be detached even though other attached packages depend on it? By default, `force = FALSE`
`keep_kim`	logical. If `keep_kim = FALSE`, Package 'kim' will be detached along with all other user-installed packages. If `keep_kim = TRUE`, Package 'kim' will not be detached. By default, `keep_kim = FALSE`

Examples

```
## Not run:
detach_user_installed_pkgs()

## End(Not run)
```

`duplicated_values` *Duplicated values in a vector*

Description

Return all duplicated values in a vector. This function is a copy of the earlier function, find_duplicates, in Package 'kim'

Usage

```
duplicated_values(vector = NULL, na.rm = TRUE, sigfigs = 2, output = "summary")
```

Arguments

`vector`	a vector whose elements will be checked for duplicates
`na.rm`	logical. If `na.rm = TRUE`, NA values in the vector will be removed before searching for duplicates. If `na.rm = FALSE`, NA values will be included in the search as potentially duplicated values. By default, `na.rm = TRUE`.
`sigfigs`	number of significant digits to round to in the percent column of the summary (default = 2)
`output`	type of output. If `output = "summary"`, the function's output will be a data.table summarizing duplicated values and their counts. If `output = "duplicated_values"`, the function's output will be a vector of duplicated values. If `output = "non_duplicated_values"`, the function's output will be a vector of non-duplicated values (default = "summary")

Value

the output will be a data.table object (summary), a vector of duplicated values, or a vector non-duplicated values.

Examples

```
duplicated_values(mtcars$cyl)
duplicated_values(mtcars$cyl, output = "duplicated_values")
duplicated_values(vector = c(mtcars$cyl, 11:20, NA, NA))
duplicated_values(vector = c(mtcars$cyl, 11:20, NA, NA), na.rm = FALSE)
duplicated_values(vector = c(mtcars$cyl, 11:20, NA, NA),
na.rm = FALSE, sigfigs = 4, output = "duplicated_values")
```

excel_formula_convert *Excel formula, convert (to)*

Description

Alias for the 'convert_to_excel_formula' function. Convert elements of a character vector to Excel formulas to preserve the character (string) format when opened in an Excel file.

Usage

```
excel_formula_convert(vector = NULL)
```

Arguments

`vector`	a character vector

Value

the output will be a character vector formatted as an Excel formula. For example, if an element in the input vector was `".500"`, this element will be converted to `=".500"`, which will show up as ".500" in Excel, rather than as "0.5"

Examples

```
## Not run:
# compare the two csv files below
# example 1
dt <- data.table::data.table(a = ".500")
data.table::fwrite(dt, "example1.csv") # the csv will show "0.5"
# example 2
dt <- data.table::data.table(a = excel_formula_convert(".500"))
data.table::fwrite(dt, "example2.csv") # the csv will show ".500"

## End(Not run)
```

exit_from_parent_function *Exit from a Parent Function*

Description

Exit from a Parent Function

Usage

```
exit_from_parent_function(
  n = 1,
  silent = FALSE,
  message = "Exiting from a parent function"
)
```

Arguments

`n`	the number of generations to go back (default = 1)
`silent`	logical. If `silent` = `TRUE`, a message will be printed.
`message`	message to print

Examples

```
fn1 <- function() {
print(1)
print(2)
}
fn1()
fn2 <- function() {
print(1)
exit_from_parent_function()
print(2)
}
fn2()
```

`factorial_anova_2_way` *Factorial ANOVA 2-Way (Two-Way Factorial ANOVA)*

Description

Conduct a two-way factorial analysis of variance (ANOVA).

Usage

```
factorial_anova_2_way(
  data = NULL,
  dv_name = NULL,
  iv_1_name = NULL,
  iv_2_name = NULL,
  iv_1_values = NULL,
  iv_2_values = NULL,
  sigfigs = 3,
  robust = FALSE,
  iterations = 2000,
  plot = TRUE,
  error_bar = "ci",
  error_bar_range = 0.95,
  error_bar_tip_width = 0.13,
  error_bar_thickness = 1,
  error_bar_caption = TRUE,
  line_colors = c("red", "blue"),
  line_types = NULL,
  line_thickness = 1,
  dot_size = 3,
  position_dodge = 0.13,
  x_axis_title = NULL,
  y_axis_title = NULL,
  y_axis_title_vjust = 0.85,
  legend_title = NULL,
  legend_position = "right",
  output = "anova_table",
  png_name = NULL,
  width = 7000,
  height = 4000,
  units = "px",
  res = 300,
  layout_matrix = NULL
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`dv_name`	name of the dependent variable
`iv_1_name`	name of the first independent variable
`iv_2_name`	name of the second independent variable
`iv_1_values`	restrict all analyses to observations having these values for the first independent variable
`iv_2_values`	restrict all analyses to observations having these values for the second independent variable
`sigfigs`	number of significant digits to which to round values in anova table (default = 3)
`robust`	if `TRUE`, conduct a robust ANOVA in addition.
`iterations`	number of bootstrap samples for robust ANOVA. The default is set at 2000, but consider increasing the number of samples to 5000, 10000, or an even larger number, if slower handling time is not an issue.
`plot`	if `TRUE`, print a plot and enable returning an output (default = TRUE)
`error_bar`	if `error_bar = "se"`; error bars will be +/-1 standard error; if `error_bar = "ci"` error bars will be a confidence interval
`error_bar_range`	width of the confidence interval (default = 0.95 for 95 percent confidence interval). This argument will not apply when `error_bar = "se"`

`error_bar_tip_width`	graphically, width of the segments at the end of error bars (default = 0.13)
`error_bar_thickness`	thickness of the error bars (default = 1)
`error_bar_caption`	should a caption be included to indicate the width of the error bars? (default = TRUE).
`line_colors`	colors of the lines connecting means (default = NULL) If the second IV has two levels, then by default, `line_colors = c("red", "blue")`
`line_types`	types of the lines connecting means (default = NULL) If the second IV has two levels, then by default, `line_types = c("solid", "dashed")`
`line_thickness`	thickness of the lines connecting group means, (default = 1)
`dot_size`	size of the dots indicating group means (default = 3)
`position_dodge`	by how much should the group means and error bars be horizontally offset from each other so as not to overlap? (default = 0.13)
`x_axis_title`	a character string for the x-axis title. If no input is entered, then, by default, the first value of `iv_name` will be used as the x-axis title.
`y_axis_title`	a character string for the y-axis title. If no input is entered, then, by default, `dv_name` will be used as the y-axis title.
`y_axis_title_vjust`	position of the y axis title (default = 0.85). By default, `y_axis_title_vjust = 0.85`, which means that the y axis title will be positioned at 85% of the way up from the bottom of the plot.
`legend_title`	a character for the legend title. If no input is entered, then, by default, the second value of `iv_name` will be used as the legend title. If `legend_title = FALSE`, then the legend title will be removed.
`legend_position`	position of the legend: `"none"`, `"top"`, `"right"`, `"bottom"`, `"left"`, `"none"` (default = `"right"`)
`output`	output type can be one of the following: `"anova_table"`, `"group_stats"`, `"plot"`, `"robust_anova_results"`, `"robust_anova_post_hoc_results"`, `"robust_anova_post_hoc` `"all"`
`png_name`	name of the PNG file to be saved. If `png_name = TRUE`, the name will be "factorial_anova_2_way_" followed by a timestamp of the current time. The timestamp will be in the format, jan_01_2021_1300_10_000001, where "jan_01_2021" would indicate January 01, 2021; 1300 would indicate 13:00 (i.e., 1 PM); and 10_000001 would indicate 10.000001 seconds after the hour.
`width`	width of the PNG file (default = 7000)
`height`	height of the PNG file (default = 4000)
`units`	the units for the `width` and `height` arguments. Can be `"px"` (pixels), `"in"` (inches), `"cm"`, or `"mm"`. By default, `units = "px"`.
`res`	The nominal resolution in ppi which will be recorded in the png file, if a positive integer. Used for units other than the default. If not specified, taken as 300 ppi to set the size of text and line widths.
`layout_matrix`	The layout argument for arranging plots and tables using the `grid.arrange` function.

Details

The following package(s) must be installed prior to running this function: Package 'car' v3.0.9 (or possibly a higher version) by Fox et al. (2020), https://cran.r-project.org/package=car

If robust ANOVA is to be conducted, the following package(s) must be installed prior to running the function: Package 'WRS2' v1.1-1 (or possibly a higher version) by Mair & Wilcox (2021), https://cran.r-project.org/package=WRS2

Value

by default, the output will be `"anova_table"`

Examples

```
factorial_anova_2_way(
  data = mtcars, dv_name = "mpg", iv_1_name = "vs",
  iv_2_name = "am", iterations = 100)
anova_results <- factorial_anova_2_way(
  data = mtcars, dv_name = "mpg", iv_1_name = "vs",
  iv_2_name = "am", output = "all")
anova_results
```

`find_duplicates` *Find duplicated values in a vector*

Description

Find duplicated values in a vector

Usage

```
find_duplicates(vector = NULL, na.rm = TRUE, sigfigs = 2, output = "summary")
```

Arguments

`vector`	a vector whose elements will be checked for duplicates
`na.rm`	logical. If `na.rm` = `TRUE`, NA values in the vector will be removed before searching for duplicates. If `na.rm` = `FALSE`, NA values will be included in the search as potentially duplicated values. By default, `na.rm` = `TRUE`.
`sigfigs`	number of significant digits to round to in the percent column of the summary (default = 2)
`output`	type of output. If `output = "summary"`, the function's output will be a data.table summarizing duplicated values and their counts. If `output = "duplicated_values"`, the function's output will be a vector of duplicated values. If `output = "non_duplicated_values"`, the function's output will be a vector of non-duplicated values (default = "summary")

Value

the output will be a data.table object (summary), a vector of duplicated values, or a vector non-duplicated values.

Examples

```
find_duplicates(mtcars$cyl)
find_duplicates(mtcars$cyl, output = "duplicated_values")
find_duplicates(vector = c(mtcars$cyl, 11:20, NA, NA))
find_duplicates(vector = c(mtcars$cyl, 11:20, NA, NA), na.rm = FALSE)
find_duplicates(vector = c(mtcars$cyl, 11:20, NA, NA),
na.rm = FALSE, sigfigs = 4, output = "duplicated_values")
```

`fisher_z_transform` *Fisher's Z transformation*

Description

Perform Fisher's r-to-Z transformation for given correlation coefficient(s).

Usage

```
fisher_z_transform(r = NULL)
```

Arguments

`r`	a (vector of) correlation coefficient(s)

Value

the output will be a vector of Z values which were transformed from the given r values.

Examples

```
fisher_z_transform(0.99)
fisher_z_transform(r = seq(0.1, 0.5, 0.1))
```

floodlight_2_by_continuous

Floodlight 2 by Continuous

Description

Conduct a floodlight analysis for 2 x Continuous design.

Usage

```
floodlight_2_by_continuous(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  mod_name = NULL,
  covariate_name = NULL,
  interaction_p_include = TRUE,
  iv_level_order = NULL,
  output = "reg_lines_plot",
  jitter_x_y_percent = 0,
  jitter_x_percent = 0,
  jitter_y_percent = 0,
  dot_alpha = 0.5,
  dot_size = 4,
  interaction_p_value_font_size = 8,
  jn_point_label_add = TRUE,
  jn_point_font_size = 8,
  jn_point_label_hjust = NULL,
  lines_at_mod_extremes = FALSE,
  interaction_p_vjust = -3,
  plot_margin = ggplot2::unit(c(75, 7, 7, 7), "pt"),
  legend_position = "right",
  reg_line_types = c("solid", "dashed"),
  jn_line_types = c("solid", "solid"),
  jn_line_thickness = 1.5,
  colors_for_iv = c("red", "blue"),
  sig_region_color = "green",
  sig_region_alpha = 0.08,
  nonsig_region_color = "gray",
  nonsig_region_alpha = 0.08,
  x_axis_title = NULL,
  y_axis_title = NULL,
  legend_title = NULL,
  round_decimals_int_p_value = 3,
  line_of_fit_thickness = 1,
  round_jn_point_labels = 2
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name`	name of the binary independent variable
`dv_name`	name of the dependent variable
`mod_name`	name of the continuous moderator variable
`covariate_name`	name of the variables to control for
`interaction_p_include`	logical. Should the plot include a p-value for the interaction term?
`iv_level_order`	order of levels in the independent variable for legend. By default, it will be set as levels of the independent variable ordered using R's base function `sort`.
`output`	type of output (default = "reg_lines_plot"). Possible inputs: "interactions_pkg_results", "simple_effects_plot", "jn_points", "regions", "reg_lines_plot"
`jitter_x_y_percent`	horizontally and vertically jitter dots by a percentage of the respective ranges of x and y values.
`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
`dot_alpha`	opacity of the dots (0 = completely transparent, 1 = completely opaque). By default, `dot_alpha = 0.5`
`dot_size`	size of the dots (default = 4)
`interaction_p_value_font_size`	font size for the interaction p value (default = 8)
`jn_point_label_add`	logical. Should the labels for Johnson-Neyman point labels be added to the plot? (default = TRUE)
`jn_point_font_size`	font size for Johnson-Neyman point labels (default = 8)
`jn_point_label_hjust`	a vector of hjust values for Johnson-Neyman point labels. By default, the hjust value will be 0.5 for all the points.
`lines_at_mod_extremes`	logical. Should vertical lines be drawn at the observed extreme values of the moderator if those values lie in siginificant region(s)? (default = FALSE)
`interaction_p_vjust`	By how much should the label for the interaction p-value be adjusted vertically? By default, `interaction_p_vjust = -3`)
`plot_margin`	margin for the plot By default `plot_margin = ggplot2::unit(c(75, 7, 7, 7), "pt")`
`legend_position`	position of the legend (default = "right"). If `legend_position = "none"`, the legend will be removed.

`reg_line_types`	types of the regression lines for the two levels of the independent variable. By default, `reg_line_types = c("solid", "dashed")`
`jn_line_types`	types of the lines for Johnson-Neyman points. By default, `jn_line_types = c("solid", "solid")`
`jn_line_thickness`	thickness of the lines at Johnson-Neyman points (default = 1.5)
`colors_for_iv`	colors for the two values of the independent variable (default = c("red", "blue"))
`sig_region_color`	color of the significant region, i.e., range(s) of the moderator variable for which simple effect of the independent variable on the dependent variable is statistically significant.
`sig_region_alpha`	opacity for `sig_region_color`. (0 = completely transparent, 1 = completely opaque). By default, `sig_region_alpha = 0.08`
`nonsig_region_color`	color of the non-significant region, i.e., range(s) of the moderator variable for which simple effect of the independent variable on the dependent variable is not statistically significant.
`nonsig_region_alpha`	opacity for `nonsig_region_color`. (0 = completely transparent, 1 = completely opaque). By default, `nonsig_region_alpha = 0.08`
`x_axis_title`	title of the x axis. By default, it will be set as input for `mod_name`. If `x_axis_title = FALSE`, it will be removed.
`y_axis_title`	title of the y axis. By default, it will be set as input for `dv_name`. If `y_axis_title = FALSE`, it will be removed.
`legend_title`	title of the legend. By default, it will be set as input for `iv_name`. If `legend_title = FALSE`, it will be removed.
`round_decimals_int_p_value`	To how many digits after the decimal point should the p value for the interaction term be rounded? (default = 3)
`line_of_fit_thickness`	thickness of the lines of fit (default = 1)
`round_jn_point_labels`	To how many digits after the decimal point should the jn point labels be rounded? (default = 2)

Details

The following package(s) must be installed prior to running this function: Package 'interactions' v1.1.1 (or possibly a higher version) by Jacob A. Long (2020), `https://cran.r-project.org/package=interactions` See the following references: Spiller et al. (2013) doi:10.1509/jmr.12.0420 Kim (2021) doi:10.5281/zenodo.4445388

Examples

```
# typical example
floodlight_2_by_continuous(
data = mtcars,
iv_name = "am",
dv_name = "mpg",
mod_name = "qsec")
# add covariates
floodlight_2_by_continuous(
data = mtcars,
iv_name = "am",
dv_name = "mpg",
mod_name = "qsec",
covariate_name = c("cyl", "hp"))
# adjust the jn point label positions
floodlight_2_by_continuous(
data = mtcars,
iv_name = "am",
dv_name = "mpg",
mod_name = "qsec",
jn_point_label_hjust = c(1, 0))
# return regions of significance and nonsignificance
floodlight_2_by_continuous(
data = mtcars,
iv_name = "am",
dv_name = "mpg",
mod_name = "qsec",
output = "regions")
# draw lines at the extreme values of the moderator
# if they are included in the significant region
floodlight_2_by_continuous(
data = mtcars,
iv_name = "am",
dv_name = "mpg",
mod_name = "qsec",
lines_at_mod_extremes = TRUE)
#' # remove the labels for jn points
floodlight_2_by_continuous(
data = mtcars,
iv_name = "am",
dv_name = "mpg",
mod_name = "qsec",
jn_point_label_add = FALSE)
```

`floodlight_2_by_continuous_logistic`
Floodlight 2 by Continuous for a Logistic Regression

Description

Conduct a floodlight analysis for a logistic regression with a 2 x Continuous design involving a binary dependent variable.

Usage

```
floodlight_2_by_continuous_logistic(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  mod_name = NULL,
  interaction_p_include = TRUE,
  iv_level_order = NULL,
  dv_level_order = NULL,
  jn_points_disregard_threshold = NULL,
  output = "reg_lines_plot",
  num_of_spotlights = 20,
  jitter_x_percent = 0,
  jitter_y_percent = 5,
  dot_alpha = 0.3,
  dot_size = 6,
  interaction_p_value_font_size = 8,
  jn_point_label_add = TRUE,
  jn_point_font_size = 8,
  jn_point_label_hjust = NULL,
  interaction_p_vjust = -3,
  plot_margin = ggplot2::unit(c(75, 7, 7, 7), "pt"),
  legend_position = "right",
  line_types_for_pred_values = c("solid", "dashed"),
  line_thickness_for_pred_values = 2.5,
  jn_line_types = c("solid", "solid"),
  jn_line_thickness = 1.5,
  sig_region_color = "green",
  sig_region_alpha = 0.08,
  nonsig_region_color = "gray",
  nonsig_region_alpha = 0.08,
  x_axis_title = NULL,
  y_axis_title = NULL,
  legend_title = NULL,
  round_decimals_int_p_value = 3,
  round_jn_point_labels = 2
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name`	name of the binary independent variable
`dv_name`	name of the binary dependent variable

`mod_name`	name of the continuous moderator variable
`interaction_p_include`	logical. Should the plot include a p-value for the interaction term?
`iv_level_order`	order of levels in the independent variable for legend. By default, it will be set as levels of the independent variable ordered using R's base function `sort`.
`dv_level_order`	order of levels in the dependent variable. By default, it will be set as levels of the dependent variable ordered using R's base function `sort`.
`jn_points_disregard_threshold`	the Minimum Distance in the unit of the moderator variable that will be used for various purposes, such as (1) to disregard the second Johnson-Neyman point that is different from the first Johnson-Neyman (JN) point by less than the Minimum Distance; (2) to determine regions of significance, which will calculate the p-value of the IV's effect (the focal dummy variable's effect) on DV at a candidate JN point + / - the Minimum Distance. This input is hard to explain, but a user can enter a really low value for this argument (e.g., `jn_points_disregard_threshold = 0.1` for a moderator measured on a 100-point scale) or use the default. By default, `jn_points_disregard_threshold = range of the moderator / 10000` For example, if the observed moderator values range from 1 to 7 (because it is a 7-point scale), then `jn_points_disregard_threshold = (7 - 1) / 10000 = 0.0006`
`output`	type of output (default = "reg_lines_plot"). Possible inputs: "interactions_pkg_results", "simple_effects_plot", "jn_points", "regions", "reg_lines_plot"
`num_of_spotlights`	How many spotlight analyses should be conducted to plot the predicted values at various values of the moderator? (default = 20)
`jitter_x_percent`	horizontally jitter dots by a percentage of the range of x values (default = 0)
`jitter_y_percent`	vertically jitter dots by a percentage of the range of y values (default = 5)
`dot_alpha`	opacity of the dots (0 = completely transparent, 1 = completely opaque). By default, `dot_alpha = 0.3`
`dot_size`	size of the dots (default = 6)
`interaction_p_value_font_size`	font size for the interaction p value (default = 8)
`jn_point_label_add`	logical. Should the labels for Johnson-Neyman point labels be added to the plot? (default = TRUE)
`jn_point_font_size`	font size for Johnson-Neyman point labels (default = 8)
`jn_point_label_hjust`	a vector of hjust values for Johnson-Neyman point labels. By default, the hjust value will be 0.5 for all the points.
`interaction_p_vjust`	By how much should the label for the interaction p-value be adjusted vertically? By default, `interaction_p_vjust = -3`)

`plot_margin`	margin for the plot By default `plot_margin = ggplot2::unit(c(75, 7, 7, 7), "pt")`
`legend_position`	position of the legend (default = "right"). If `legend_position = "none"`, the legend will be removed.
`line_types_for_pred_values`	types of the lines for plotting the predicted values By default, `line_types_for_pred_values = c("solid", "dashed")`
`line_thickness_for_pred_values`	thickness of the lines for plotting the predicted values (default = 2.5)
`jn_line_types`	types of the lines for Johnson-Neyman points. By default, `jn_line_types = c("solid", "solid")`
`jn_line_thickness`	thickness of the lines at Johnson-Neyman points (default = 1.5)
`sig_region_color`	color of the significant region, i.e., range(s) of the moderator variable for which simple effect of the independent variable on the dependent variable is statistically significant.
`sig_region_alpha`	opacity for `sig_region_color`. (0 = completely transparent, 1 = completely opaque). By default, `sig_region_alpha = 0.08`
`nonsig_region_color`	color of the non-significant region, i.e., range(s) of the moderator variable for which simple effect of the independent variable on the dependent variable is not statistically significant.
`nonsig_region_alpha`	opacity for `nonsig_region_color`. (0 = completely transparent, 1 = completely opaque). By default, `nonsig_region_alpha = 0.08`
`x_axis_title`	title of the x axis. By default, it will be set as input for `mod_name`. If `x_axis_title = FALSE`, it will be removed.
`y_axis_title`	title of the y axis. By default, it will be set as input for `dv_name`. If `y_axis_title = FALSE`, it will be removed.
`legend_title`	title of the legend. By default, it will be set as input for `iv_name`. If `legend_title = FALSE`, it will be removed.
`round_decimals_int_p_value`	To how many digits after the decimal point should the p value for the interaction term be rounded? (default = 3)
`round_jn_point_labels`	To how many digits after the decimal point should the jn point labels be rounded? (default = 2)

Details

See the following reference(s): Spiller et al. (2013) doi:10.1509/jmr.12.0420 Kim (2023) `https://jinkim.science/docs/floodlight.pdf`

Examples

```
floodlight_2_by_continuous_logistic(
data = mtcars,
iv_name = "am",
dv_name = "vs",
mod_name = "mpg")
# adjust the number of spotlights
# (i.e., predict values at only 4 values of the moderator)
floodlight_2_by_continuous_logistic(
data = mtcars,
iv_name = "am",
dv_name = "vs",
mod_name = "mpg",
num_of_spotlights = 4)
```

---

floodlight_2_by_continuous_mlm_logistic
Floodlight 2 by Continuous for a Multilevel Logistic Regression

---

Description

Conduct a floodlight analysis for a multilevellogistic regression with a 2 x Continuous design involving a binary dependent variable.

Usage

```
floodlight_2_by_continuous_mlm_logistic(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  mod_name = NULL,
  interaction_p_include = TRUE,
  iv_level_order = NULL,
  dv_level_order = NULL,
  jn_points_disregard_threshold = NULL,
  output = "reg_lines_plot",
  num_of_spotlights = 20,
  jitter_x_percent = 0,
  jitter_y_percent = 5,
  dot_alpha = 0.3,
  dot_size = 6,
  interaction_p_value_font_size = 8,
  jn_point_font_size = 8,
  jn_point_label_hjust = NULL,
  interaction_p_vjust = -3,
  plot_margin = ggplot2::unit(c(75, 7, 7, 7), "pt"),
  legend_position = "right",
  line_types_for_pred_values = c("solid", "dashed"),
  line_thickness_for_pred_values = 2.5,
  jn_line_types = c("solid", "solid"),
  jn_line_thickness = 1.5,
  sig_region_color = "green",
  sig_region_alpha = 0.08,
  nonsig_region_color = "gray",
  nonsig_region_alpha = 0.08,
  x_axis_title = NULL,
  y_axis_title = NULL,
  legend_title = NULL,
  round_decimals_int_p_value = 3,
  round_jn_point_labels = 2
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name`	name of the binary independent variable
`dv_name`	name of the binary dependent variable
`mod_name`	name of the continuous moderator variable
`interaction_p_include`	logical. Should the plot include a p-value for the interaction term?
`iv_level_order`	order of levels in the independent variable for legend. By default, it will be set as levels of the independent variable ordered using R's base function `sort`.
`dv_level_order`	order of levels in the dependent variable. By default, it will be set as levels of the dependent variable ordered using R's base function `sort`.
`jn_points_disregard_threshold`	the Minimum Distance in the unit of the moderator variable that will be used for various purposes, such as (1) to disregard the second Johnson-Neyman point that is different from the first Johnson-Neyman (JN) point by less than the Minimum Distance; (2) to determine regions of significance, which will calculate the p-value of the IV's effect (the focal dummy variable's effect) on DV at a candidate JN point + / - the Minimum Distance. This input is hard to explain, but a user can enter a really low value for this argument (e.g., `jn_points_disregard_threshold = 0.1` for a moderator measured on a 100-point scale) or use the default. By default, `jn_points_disregard_threshold = range of the moderator / 10000` For example, if the observed moderator values range from 1 to 7 (because it is a 7-point scale), then `jn_points_disregard_threshold = (7 - 1) / 10000 = 0.0006`
`output`	type of output (default = "reg_lines_plot"). Possible inputs: "interactions_pkg_results", "simple_effects_plot", "jn_points", "regions", "reg_lines_plot"
`num_of_spotlights`	How many spotlight analyses should be conducted to plot the predicted values at various values of the moderator? (default = 20)

`jitter_x_percent`	horizontally jitter dots by a percentage of the range of x values (default = 0)
`jitter_y_percent`	vertically jitter dots by a percentage of the range of y values (default = 5)
`dot_alpha`	opacity of the dots (0 = completely transparent, 1 = completely opaque). By default, `dot_alpha = 0.3`
`dot_size`	size of the dots (default = 6)
`interaction_p_value_font_size`	font size for the interaction p value (default = 8)
`jn_point_font_size`	font size for Johnson-Neyman point labels (default = 8)
`jn_point_label_hjust`	a vector of hjust values for Johnson-Neyman point labels. By default, the hjust value will be 0.5 for all the points.
`interaction_p_vjust`	By how much should the label for the interaction p-value be adjusted vertically? By default, `interaction_p_vjust = -3`)
`plot_margin`	margin for the plot By default `plot_margin = ggplot2::unit(c(75, 7, 7, 7), "pt")`
`legend_position`	position of the legend (default = "right"). If `legend_position = "none"`, the legend will be removed.
`line_types_for_pred_values`	types of the lines for plotting the predicted values By default, `line_types_for_pred_values = c("solid", "dashed")`
`line_thickness_for_pred_values`	thickness of the lines for plotting the predicted values (default = 2.5)
`jn_line_types`	types of the lines for Johnson-Neyman points. By default, `jn_line_types = c("solid", "solid")`
`jn_line_thickness`	thickness of the lines at Johnson-Neyman points (default = 1.5)
`sig_region_color`	color of the significant region, i.e., range(s) of the moderator variable for which simple effect of the independent variable on the dependent variable is statistically significant.
`sig_region_alpha`	opacity for `sig_region_color`. (0 = completely transparent, 1 = completely opaque). By default, `sig_region_alpha = 0.08`
`nonsig_region_color`	color of the non-significant region, i.e., range(s) of the moderator variable for which simple effect of the independent variable on the dependent variable is not statistically significant.
`nonsig_region_alpha`	opacity for `nonsig_region_color`. (0 = completely transparent, 1 = completely opaque). By default, `nonsig_region_alpha = 0.08`

`x_axis_title`	title of the x axis. By default, it will be set as input for `mod_name`. If `x_axis_title = FALSE`, it will be removed.
`y_axis_title`	title of the y axis. By default, it will be set as input for `dv_name`. If `y_axis_title = FALSE`, it will be removed.
`legend_title`	title of the legend. By default, it will be set as input for `iv_name`. If `legend_title = FALSE`, it will be removed.
`round_decimals_int_p_value`	To how many digits after the decimal point should the p value for the interaction term be rounded? (default = 3)
`round_jn_point_labels`	To how many digits after the decimal point should the jn point labels be rounded? (default = 2)

Details

See the following reference(s): Spiller et al. (2013) doi:10.1509/jmr.12.0420 Kim (2023) `https://jinkim.science/docs/floodlight.pdf`

Examples

```
floodlight_2_by_continuous_logistic(
data = mtcars,
iv_name = "am",
dv_name = "vs",
mod_name = "mpg")
# adjust the number of spotlights
# (i.e., predict values at only 4 values of the moderator)
floodlight_2_by_continuous_logistic(
data = mtcars,
iv_name = "am",
dv_name = "vs",
mod_name = "mpg",
num_of_spotlights = 4)
```

`floodlight_for_contrasts`

Floodlight Analyses for a Set of Contrasts

Description

Conduct a floodlight analysis for a set of contrasts with a continuous moderator variable.

Usage

```
floodlight_for_contrasts(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  mod_name = NULL,
  contrasts = NULL,
  contrasts_for_floodlight = NULL,
  covariate_name = NULL,
  interaction_p_include = TRUE,
  iv_category_order = NULL,
  heteroskedasticity_consistent_se = "HC4",
  round_r_squared = 3,
  round_f = 2,
  sigfigs = 2,
  jn_points_disregard_threshold = NULL,
  print_floodlight_plots = TRUE,
  output = "reg_lines_plot",
  jitter_x_percent = 0,
  jitter_y_percent = 0,
  dot_alpha = 0.5,
  dot_size = 4,
  interaction_p_value_font_size = 6,
  jn_point_font_size = 6,
  jn_point_label_hjust = NULL,
  interaction_p_vjust = -3,
  plot_margin = ggplot2::unit(c(75, 7, 7, 7), "pt"),
  legend_position = "right",
  line_of_fit_types = c("solid", "dashed"),
  line_of_fit_thickness = 1.5,
  jn_line_types = c("solid", "solid"),
  jn_line_thickness = 1.5,
  sig_region_color = "green",
  sig_region_alpha = 0.08,
  nonsig_region_color = "gray",
  nonsig_region_alpha = 0.08,
  x_axis_title = NULL,
  y_axis_title = NULL,
  legend_title = NULL,
  round_decimals_int_p_value = 3,
  round_jn_point_labels = 2
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name`	name of the multicategorical independent variable; this variable must have three or more categories.

`dv_name`	name of the dependent variable
`mod_name`	name of the continuous moderator variable
`contrasts`	names of the contrast variables
`contrasts_for_floodlight`	names of the contrast variables for which floodlight analyses will be conducted
`covariate_name`	name of the variables to control for
`interaction_p_include`	logical. Should the plot include a p-value for the interaction term?
`iv_category_order`	order of levels in the independent variable for legend. By default, it will be set as levels of the independent variable ordered using R's base function `sort`.
`heteroskedasticity_consistent_se`	which kind of heteroskedasticity-consistent (robust) standard errors should be calculated? (default = "HC4")
`round_r_squared`	number of decimal places to which to round r-squared values (default = 3)
`round_f`	number of decimal places to which to round the f statistic for model comparison (default = 2)
`sigfigs`	number of significant digits to round to (for values in the regression tables, except for p values). By default `sigfigs = 2`
`jn_points_disregard_threshold`	the Minimum Distance in the unit of the moderator variable that will be used for various purposes, such as (1) to disregard the second Johnson-Neyman point that is different from the first Johnson-Neyman (JN) point by less than the Minimum Distance; (2) to determine regions of significance, which will calculate the p-value of the IV's effect (the focal dummy variable's effect) on DV at a candidate JN point + / - the Minimum Distance. This input is hard to explain, but a user can enter a really low value for this argument (e.g., `jn_points_disregard_threshold = 0.1` for a moderator measured on a 100-point scale) or use the default. By default, `jn_points_disregard_threshold = range of the moderator / 10000` For example, if the observed moderator values range from 1 to 7 (because it is a 7-point scale), then `jn_points_disregard_threshold = (7 - 1) / 10000 = 0.0006`
`print_floodlight_plots`	If `print_floodlight_plots = TRUE`, a floodlight plot for each dummy variable will be printed. By default, `print_floodlight_plots = TRUE`
`output`	output of the function (default = "all"). Possible inputs: "reg_models", "reg_tables", "reg_tables_rounded", "all"
`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
`dot_alpha`	opacity of the dots (0 = completely transparent, 1 = completely opaque). By default, `dot_alpha = 0.5`
`dot_size`	size of the dots (default = 4)

`interaction_p_value_font_size`	font size for the interaction p value (default = 8)
`jn_point_font_size`	font size for Johnson-Neyman point labels (default = 6)
`jn_point_label_hjust`	a vector of hjust values for Johnson-Neyman point labels. By default, the hjust value will be 0.5 for all the points.
`interaction_p_vjust`	By how much should the label for the interaction p-value be adjusted vertically? By default, `interaction_p_vjust = -3`)
`plot_margin`	margin for the plot By default `plot_margin = ggplot2::unit(c(75, 7, 7, 7), "pt")`
`legend_position`	position of the legend (default = "right"). If `legend_position = "none"`, the legend will be removed.
`line_of_fit_types`	types of the lines of fit for the two levels of the independent variable. By default, `line_of_fit_types = c("solid", "dashed")`
`line_of_fit_thickness`	thickness of the lines of fit (default = 1.5)
`jn_line_types`	types of the lines for Johnson-Neyman points. By default, `jn_line_types = c("solid", "solid")`
`jn_line_thickness`	thickness of the lines at Johnson-Neyman points (default = 1.5)
`sig_region_color`	color of the significant region, i.e., range(s) of the moderator variable for which simple effect of the independent variable on the dependent variable is statistically significant.
`sig_region_alpha`	opacity for `sig_region_color`. (0 = completely transparent, 1 = completely opaque). By default, `sig_region_alpha = 0.08`
`nonsig_region_color`	color of the non-significant region, i.e., range(s) of the moderator variable for which simple effect of the independent variable on the dependent variable is not statistically significant.
`nonsig_region_alpha`	opacity for `nonsig_region_color`. (0 = completely transparent, 1 = completely opaque). By default, `nonsig_region_alpha = 0.08`
`x_axis_title`	title of the x axis. By default, it will be set as input for `mod_name`. If `x_axis_title = FALSE`, it will be removed.
`y_axis_title`	title of the y axis. By default, it will be set as input for `dv_name`. If `y_axis_title = FALSE`, it will be removed.
`legend_title`	title of the legend. By default, it will be set as input for `iv_name`. If `legend_title = FALSE`, it will be removed.

`round_decimals_int_p_value`
: To how many digits after the decimal point should the p value for the interaction term be rounded? (default = 3)

`round_jn_point_labels`
: To how many digits after the decimal point should the jn point labels be rounded? (default = 2)

Details

See the following reference, which covers a related topic: Hayes & Montoya (2017) doi:10.1080/19312458.2016.1271116

Examples

```
## Not run:
# typical example
# copy and modify the 'mtcars' data
mtcars2 <- setDT(data.table::copy(mtcars))
# make sure the data table package is attached
mtcars2[, contrast_1 := fcase(cyl == 4, -2, cyl %in% c(6, 8), 1)]
mtcars2[, contrast_2 := fcase(cyl == 4, 0, cyl == 6, 1, cyl == 8, -1)]
floodlight_for_contrasts(
data = mtcars2,
iv_name = "cyl",
dv_name = "mpg",
mod_name = "qsec",
contrasts = paste0("contrast_", 1:2),
contrasts_for_floodlight = "contrast_2")

## End(Not run)
```

`floodlight_multi_by_continuous`
Floodlight Multicategorical by Continuous

Description

Conduct a floodlight analysis for a Multicategorical IV x Continuous Moderator design.

Usage

```
floodlight_multi_by_continuous(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  mod_name = NULL,
  coding = "indicator",
  baseline_category = NULL,
  covariate_name = NULL,
  interaction_p_include = TRUE,
  iv_category_order = NULL,
  heteroskedasticity_consistent_se = "HC4",
  round_r_squared = 3,
  round_f = 2,
  sigfigs = 2,
  jn_points_disregard_threshold = NULL,
  print_floodlight_plots = TRUE,
  output = "all",
  jitter_x_percent = 0,
  jitter_y_percent = 0,
  dot_alpha = 0.5,
  dot_size = 4,
  interaction_p_value_font_size = 8,
  jn_point_font_size = 8,
  jn_point_label_hjust = NULL,
  interaction_p_vjust = -3,
  plot_margin = ggplot2::unit(c(75, 7, 7, 7), "pt"),
  legend_position = "right",
  line_of_fit_types = c("solid", "dashed"),
  line_of_fit_thickness = 1.5,
  jn_line_types = c("solid", "solid"),
  jn_line_thickness = 1.5,
  colors_for_iv = c("red", "blue"),
  sig_region_color = "green",
  sig_region_alpha = 0.08,
  nonsig_region_color = "gray",
  nonsig_region_alpha = 0.08,
  x_axis_title = NULL,
  y_axis_title = NULL,
  legend_title = NULL,
  round_decimals_int_p_value = 3,
  round_jn_point_labels = 2
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name`	name of the multicategorical independent variable; this variable must have three or more categories.
`dv_name`	name of the dependent variable
`mod_name`	name of the continuous moderator variable
`coding`	name of the coding scheme to use; the current version of the function allows only the "indicator" coding scheme. By default, `coding = "indicator"`
`baseline_category`	value of the independent variable that will be the reference value against which other values of the independent variable will be compared

`covariate_name`	name of the variables to control for
`interaction_p_include`	logical. Should the plot include a p-value for the interaction term?
`iv_category_order`	order of levels in the independent variable for legend. By default, it will be set as levels of the independent variable ordered using R's base function `sort`.
`heteroskedasticity_consistent_se`	which kind of heteroskedasticity-consistent (robust) standard errors should be calculated? (default = "HC4")
`round_r_squared`	number of decimal places to which to round r-squared values (default = 3)
`round_f`	number of decimal places to which to round the f statistic for model comparison (default = 2)
`sigfigs`	number of significant digits to round to (for values in the regression tables, except for p values). By default `sigfigs = 2`
`jn_points_disregard_threshold`	the Minimum Distance in the unit of the moderator variable that will be used for various purposes, such as (1) to disregard the second Johnson-Neyman point that is different from the first Johnson-Neyman (JN) point by less than the Minimum Distance; (2) to determine regions of significance, which will calculate the p-value of the IV's effect (the focal dummy variable's effect) on DV at a candidate JN point + / - the Minimum Distance. This input is hard to explain, but a user can enter a really low value for this argument (e.g., `jn_points_disregard_threshold = 0.1` for a moderator measured on a 100-point scale) or use the default. By default, `jn_points_disregard_threshold = range of the moderator / 10000` For example, if the observed moderator values range from 1 to 7 (because it is a 7-point scale), then `jn_points_disregard_threshold = (7 - 1) / 10000 = 0.0006`
`print_floodlight_plots`	If `print_floodlight_plots = TRUE`, a floodlight plot for each dummy variable will be printed. By default, `print_floodlight_plots = TRUE`
`output`	output of the function (default = "all"). Possible inputs: "reg_models", "reg_tables", "reg_tables_rounded", "all"
`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
`dot_alpha`	opacity of the dots (0 = completely transparent, 1 = completely opaque). By default, `dot_alpha = 0.5`
`dot_size`	size of the dots (default = 4)
`interaction_p_value_font_size`	font size for the interaction p value (default = 8)
`jn_point_font_size`	font size for Johnson-Neyman point labels (default = 8)

`jn_point_label_hjust`	a vector of hjust values for Johnson-Neyman point labels. By default, the hjust value will be 0.5 for all the points.
`interaction_p_vjust`	By how much should the label for the interaction p-value be adjusted vertically? By default, `interaction_p_vjust = -3`)
`plot_margin`	margin for the plot By default `plot_margin = ggplot2::unit(c(75, 7, 7, 7), "pt")`
`legend_position`	position of the legend (default = "right"). If `legend_position = "none"`, the legend will be removed.
`line_of_fit_types`	types of the lines of fit for the two levels of the independent variable. By default, `line_of_fit_types = c("solid", "dashed")`
`line_of_fit_thickness`	thickness of the lines of fit (default = 1.5)
`jn_line_types`	types of the lines for Johnson-Neyman points. By default, `jn_line_types = c("solid", "solid")`
`jn_line_thickness`	thickness of the lines at Johnson-Neyman points (default = 1.5)
`colors_for_iv`	colors for the two values of the independent variable (default = c("red", "blue"))
`sig_region_color`	color of the significant region, i.e., range(s) of the moderator variable for which simple effect of the independent variable on the dependent variable is statistically significant.
`sig_region_alpha`	opacity for `sig_region_color`. (0 = completely transparent, 1 = completely opaque). By default, `sig_region_alpha = 0.08`
`nonsig_region_color`	color of the non-significant region, i.e., range(s) of the moderator variable for which simple effect of the independent variable on the dependent variable is not statistically significant.
`nonsig_region_alpha`	opacity for `nonsig_region_color`. (0 = completely transparent, 1 = completely opaque). By default, `nonsig_region_alpha = 0.08`
`x_axis_title`	title of the x axis. By default, it will be set as input for `mod_name`. If `x_axis_title = FALSE`, it will be removed.
`y_axis_title`	title of the y axis. By default, it will be set as input for `dv_name`. If `y_axis_title = FALSE`, it will be removed.
`legend_title`	title of the legend. By default, it will be set as input for `iv_name`. If `legend_title = FALSE`, it will be removed.
`round_decimals_int_p_value`	To how many digits after the decimal point should the p value for the interaction term be rounded? (default = 3)
`round_jn_point_labels`	To how many digits after the decimal point should the jn point labels be rounded? (default = 2)

Details

See the following reference: Hayes & Montoya (2017) doi:10.1080/19312458.2016.1271116 Williams (2004) on r-squared values when calculating robust standard errors `https://web.archive.org/web/20230627025457/https://www.stata.com/statalist/archive/2004-05/msg00107.html`

Examples

```
## Not run:
# typical example
floodlight_multi_by_continuous(
data = mtcars,
iv_name = "cyl",
dv_name = "mpg",
mod_name = "qsec")

## End(Not run)
```

`forest_plot` *Forest plot*

Description

Create a forest plot using outputs from 'metafor' package

Usage

```
forest_plot(
  estimates = NULL,
  estimate_ci_ll = NULL,
  estimate_ci_ul = NULL,
  point_size_range = c(2, 10),
  error_bar_size = 1,
  error_bar_tip_height = 0.3,
  weights = NULL,
  diamond_x = NULL,
  diamond_ci_ll = NULL,
  diamond_ci_ul = NULL,
  diamond_height = 1.2,
  diamond_gap_height = 0.3,
  diamond_1_tip_at_top_y = -0.5,
  diamond_colors = "black",
  study_labels = NULL,
  diamond_labels = NULL,
  diamond_label_size = 6,
  diamond_label_hjust = 0,
  diamond_label_fontface = "bold",
  diamond_estimate_label_hjust = 0,
  diamond_estimate_label_size = 6,
  diamond_estimate_label_fontface = "bold",
  round_estimates = 2,
  x_axis_title = "Observed Outcome",
  vline_size = 1,
  vline_intercept = 0,
  vline_type = "dotted",
  study_label_hjust = 0,
  study_label_begin_x = NULL,
  study_label_begin_x_perc = 60,
  study_label_size = 6,
  study_label_fontface = "plain",
  estimate_label_begin_x = NULL,
  estimate_label_begin_x_perc = 25,
  estimate_label_hjust = 0,
  estimate_label_size = 6,
  estimate_label_fontface = "plain",
  x_axis_tick_marks = NULL,
  x_axis_tick_mark_label_size = 6,
  legend_position = "none",
  plot_margin = NULL
)
```

Arguments

`estimates`	default = NULL
`estimate_ci_ll`	default = NULL
`estimate_ci_ul`	default = NULL
`point_size_range`	default = c(2, 10)
`error_bar_size`	default = 1
`error_bar_tip_height`	default = 0.3
`weights`	default = NULL
`diamond_x`	default = NULL
`diamond_ci_ll`	default = NULL
`diamond_ci_ul`	default = NULL
`diamond_height`	default = 1.2
`diamond_gap_height`	default = 0.3
`diamond_1_tip_at_top_y`	default = -0.5
`diamond_colors`	default = "black"
`study_labels`	default = NULL
`diamond_labels`	default = NULL

`diamond_label_size`	default = 6
`diamond_label_hjust`	default = 0
`diamond_label_fontface`	default = "bold"
`diamond_estimate_label_hjust`	default = 0
`diamond_estimate_label_size`	default = 6
`diamond_estimate_label_fontface`	default = "bold"
`round_estimates`	default = 2
`x_axis_title`	default = "Observed Outcome"
`vline_size`	default = 1
`vline_intercept`	default = 0
`vline_type`	default = "dotted"
`study_label_hjust`	default = 0
`study_label_begin_x`	default = NULL
`study_label_begin_x_perc`	default = 60
`study_label_size`	default = 6
`study_label_fontface`	default = "plain"
`estimate_label_begin_x`	default = NULL
`estimate_label_begin_x_perc`	default = 25
`estimate_label_hjust`	default = 0
`estimate_label_size`	default = 6
`estimate_label_fontface`	default = "plain"
`x_axis_tick_marks`	default = NULL
`x_axis_tick_mark_label_size`	default = 6
`legend_position`	default = "none"
`plot_margin`	default = NULL

Examples

```
forest_plot(
estimates = c(2, 3, 4),
estimate_ci_ll = c(1, 2, 3),
estimate_ci_ul = c(3, 4, 6),
weights = 1:3,
diamond_x = 2,
diamond_labels = "RE",
diamond_ci_ll = 1.8,
diamond_ci_ul = 2.2,
estimate_label_begin_x_perc = 40,
x_axis_tick_marks = seq(-2, 6, 2))
```

geomean *Geometric mean*

Description

Calculate the geometric mean of a numeric vector

Usage

```
geomean(x = NULL, zero_or_neg_convert_to = NA)
```

Arguments

`x`	a numeric vector
`zero_or_neg_convert_to`	the value to which zero or negative values will be converted to. If `zero_or_neg_convert_to` == `NA`, zero or negative values will be converted to NA values and thus be excluded when calculating the geometric mean. (default = NA)

Examples

```
## Not run:
geomean(c(1, 4))
geomean(c(1, 100))
geomean(c(1, 100, NA))
geomean(c(1, 100, NA, 0, -1, -2))
geomean(
x = c(1, 100, NA, 0, -1, -2),
zero_or_neg_convert_to = 1)
geomean(c(1, 100, NA, 1, 1, 1))

## End(Not run)
```

ggsave_quick *ggsave quick*

Description

quickly save the current plot with a timestamp

Usage

```
ggsave_quick(
  name = NULL,
  file_name_extension = "png",
  timestamp = NULL,
  width = 16,
  height = 9
)
```

Arguments

`name`	a character string of the png file name. By default, if no input is given (`name = NULL`), the file name will begin with "ggplot". If the desired output file name is "myplot.png", enter `name = "myplot", timestamp = FALSE`
`file_name_extension`	file name extension (default = "png"). If `file_name_extension = "svg"`, Package svglite needs to be installed.
`timestamp`	if `timestamp = TRUE`, a timestamp of the current time will be appended to the file name. The timestamp will be in the format, jan_01_2021_1300_10_000001, where "jan_01_2021" would indicate January 01, 2021; 1300 would indicate 13:00 (i.e., 1 PM); and 10_000001 would indicate 10.000001 seconds after the hour. By default, `timestamp` will be set as TRUE, if no input is given for the `name` argument, and as FALSE, if an input is given for the `name` argument.
`width`	width of the plot to be saved. This argument will be directly entered as the `width` argument for the `ggsave` function within `ggplot2` package (default = 16)
`height`	height of the plot to be saved. This argument will be directly entered as the `height` argument for the `ggsave` function within `ggplot2` package (default = 9)

Value

the output will be a .png image file in the working directory.

Examples

```
## Not run:
kim::histogram(rep(1:30, 3))
ggsave_quick()

## End(Not run)
```

`histogram` *Histogram*

Description

Create a histogram based on the output of the `hist` function in the `graphics` package.

Usage

```
histogram(
  vector = NULL,
  breaks = NULL,
  counts = NULL,
  percent = FALSE,
  bin_fill_color = "green4",
  bin_border_color = "black",
  bin_border_thickness = 1,
  notify_na_count = NULL,
  x_axis_tick_marks = NULL,
  y_axis_tick_marks = NULL,
  cap_axis_lines = TRUE,
  x_axis_title = "Value",
  y_axis_title = NULL,
  y_axis_title_vjust = 0.85
)
```

Arguments

`vector`	a numeric vector
`breaks`	a numeric vector indicating breaks for the bins. By default, no input is required for this argument.
`counts`	a numeric vector containing counts for the bins (i.e., heights of the bins). By default, no input is required for this argument.
`percent`	logical. If `percent = TRUE`, percentages will be plotted rather than frequencies (default = FALSE).
`bin_fill_color`	color of the area inside each bin (default = "green4")
`bin_border_color`	color of the border around each bin (default = "black")
`bin_border_thickness`	thickness of the border around each bin (default = 1)
`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.
`x_axis_tick_marks`	a vector of values at which to place tick marks on the x axis (e.g., setting `x_axis_tick_marks = seq(0, 10, 5)` will put tick marks at 0, 5, and 10.)

`y_axis_tick_marks`	a vector of values at which to place tick marks on the y axis (e.g., setting `y_axis_tick_marks = seq(0, 10, 5)` will put tick marks at 0, 5, and 10.)
`cap_axis_lines`	logical. Should the axis lines be capped at the outer tick marks? (default = FALSE)
`x_axis_title`	title for x axis (default = "Value")
`y_axis_title`	title for y axis (default = "Count" or "Percentage", depending on the value of `percent`)
`y_axis_title_vjust`	position of the y axis title (default = 0.85).

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"]], percent = TRUE)
histogram(vector = mtcars[["mpg"]],
x_axis_tick_marks = c(10, 25, 35), 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,
  xlab = NULL,
  ylab = NULL,
  x_limits = NULL,
  x_breaks = NULL,
  x_labels = NULL,
  sigfigs = 3,
  convert_dv_to_numeric = TRUE
)
```

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)
`xlab`	title of the x-axis for the histogram by group. If `xlab = FALSE`, the title will be removed. By default (i.e., if no input is given), `dv_name` will be used as the title.
`ylab`	title of the y-axis for the histogram by group. If `ylab = FALSE`, the title will be removed. By default (i.e., if no input is given), `iv_name` will be used as the title.
`x_limits`	a numeric vector with values of the endpoints of the x axis.
`x_breaks`	a numeric vector indicating the points at which to place tick marks on the x axis.
`x_labels`	a vector containing labels for the place tick marks on the x axis.
`sigfigs`	number of significant digits to round to (default = 3)
`convert_dv_to_numeric`	logical. Should the values in the dependent variable be converted to numeric for plotting the histograms? (default = TRUE)

Details

The following package(s) must be installed prior to running this function: Package 'ggridges' v0.5.3 (or possibly a higher version) by Claus O. Wilke (2021), `https://cran.r-project.org/package=ggridges`

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", x_breaks = 4:8,
x_limits = c(4, 8))
```

---

histogram_deprecated_1 *Histogram*

Description

Create a histogram

Usage

```
histogram_deprecated_1(
  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_deprecated_1(1:100)
histogram_deprecated_1(c(1:100, NA))
histogram_deprecated_1(vector = mtcars[["mpg"]])
histogram_deprecated_1(
vector = mtcars[["mpg"]], x_tick_marks = seq(10, 36, 2))
histogram_deprecated_1(
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_from_hist` *Histogram from hist function*

Description

Create a histogram based on the output of the `hist` function in the `graphics` package.

Usage

```
histogram_from_hist(
  vector = NULL,
  breaks = NULL,
  counts = NULL,
  percent = FALSE,
  bin_fill_color = "green4",
  bin_border_color = "black",
  bin_border_thickness = 1,
  notify_na_count = NULL,
  x_axis_tick_marks = NULL,
  y_axis_tick_marks = NULL,
  cap_axis_lines = TRUE,
  x_axis_title = "Value",
  y_axis_title = NULL,
  y_axis_title_vjust = 0.85
)
```

Arguments

`vector`	a numeric vector
`breaks`	a numeric vector indicating breaks for the bins. By default, no input is required for this argument.
`counts`	a numeric vector containing counts for the bins (i.e., heights of the bins). By default, no input is required for this argument.
`percent`	logical. If `percent` = `TRUE`, percentages will be plotted rather than frequencies (default = FALSE).
`bin_fill_color`	color of the area inside each bin (default = "green4")
`bin_border_color`	color of the border around each bin (default = "black")
`bin_border_thickness`	thickness of the border around each bin (default = 1)
`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.
`x_axis_tick_marks`	a vector of values at which to place tick marks on the x axis (e.g., setting `x_axis_tick_marks` = `seq(0, 10, 5)` will put tick marks at 0, 5, and 10.)
`y_axis_tick_marks`	a vector of values at which to place tick marks on the y axis (e.g., setting `y_axis_tick_marks` = `seq(0, 10, 5)` will put tick marks at 0, 5, and 10.)
`cap_axis_lines`	logical. Should the axis lines be capped at the outer tick marks? (default = FALSE)
`x_axis_title`	title for x axis (default = "Value")
`y_axis_title`	title for y axis (default = "Count" or "Percentage", depending on the value of `percent`)
`y_axis_title_vjust`	position of the y axis title (default = 0.85).

Value

the output will be a histogram, a ggplot object.

Examples

```
histogram_from_hist(1:100)
histogram_from_hist(c(1:100, NA))
histogram_from_hist(vector = mtcars[["mpg"]])
histogram_from_hist(vector = mtcars[["mpg"]], percent = TRUE)
histogram_from_hist(vector = mtcars[["mpg"]],
x_axis_tick_marks = c(10, 25, 35), y_axis_title_vjust = 0.5,
y_axis_title = "Freq", x_axis_title = "Values of mpg")
```

`histogram_w_outlier_bins` *Histogram with outlier bins*

Description

Create a histogram with outlier bins

Usage

```
histogram_w_outlier_bins(
  vector = NULL,
  bin_cutoffs = NULL,
  outlier_bin_left = TRUE,
  outlier_bin_right = TRUE,
  x_tick_marks = NULL,
  x_tick_mark_labels = NULL,
  y_tick_marks = NULL,
  outlier_bin_fill_color = "coral",
  non_outlier_bin_fill_color = "cyan4",
  border_color = "black",
  y_axis_title_vjust = 0.85,
  x_axis_title = NULL,
  y_axis_title = NULL,
  notify_na_count = NULL,
  plot_proportion = TRUE,
  plot_frequency = FALSE,
  mean = TRUE,
  ci = TRUE,
  median = TRUE,
  median_position = 15,
  error_bar_size = 3
)
```

Arguments

`vector`	a numeric vector
`bin_cutoffs`	cutoff points for bins
`outlier_bin_left`	logical. Should the leftmost bin treated as an outlier bin? (default = TRUE)

`outlier_bin_right`	logical. Should the rightmost bin treated as an outlier bin? (default = TRUE)
`x_tick_marks`	a vector of values at which to place tick marks on the x axis. Note that the first bar spans from 0.5 to 1.5, second bar from 1.5 to 2.5, ... nth bar from n - 0.5 to n + 0.5. See the example. By default, tick marks will be placed at every cutoff point for bins
`x_tick_mark_labels`	a character vector to label tick marks. By default, the vector of cutoff points for bins will also be used as labels.
`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.)
`outlier_bin_fill_color`	color to fill inside of the outlier bins (default = "coral")
`non_outlier_bin_fill_color`	color to fill inside of the non-outlier 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"). If `x_axis_title = FALSE`, x axis title will be removed from the plot.
`y_axis_title`	title for y axis. By default, it will be either "Proportion" or "Count".
`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.
`plot_proportion`	logical. Should proportions be plotted, as opposed to frequencies? (default = TRUE)
`plot_frequency`	logical. Should frequencies be plotted, as opposed to proportions? (default = FALSE). If `plot_frequency = TRUE`, `plot_proportion` will switch to be FALSE.
`mean`	logical. Should mean marked on the histogram? (default = TRUE)
`ci`	logical. Should 95% confidence interval marked on the histogram? (default = TRUE)
`median`	logical. Should median marked on the histogram? (default = TRUE)
`median_position`	position of the median label as a percentage of height of the tallest bin (default = 15)
`error_bar_size`	size of the error bars (default = 3)

Value

a ggplot object

Examples

```
histogram_w_outlier_bins(vector = 1:100, bin_cutoffs = seq(0, 100, 10))
histogram_w_outlier_bins(vector = 0:89, bin_cutoffs = seq(0, 90, 10),
x_tick_marks = seq(0.5, 9.5, 3), x_tick_mark_labels = seq(0, 90, 30))
histogram_w_outlier_bins(vector = 1:10, bin_cutoffs = seq(0, 10, 2.5))
histogram_w_outlier_bins(vector = 1:5, bin_cutoffs = seq(0, 10, 2.5))
histogram_w_outlier_bins(vector = 1:15, bin_cutoffs = c(5.52, 10.5))
```

`holm_adjusted_p` *Holm-adjusted p-values*

Description

Adjust a vector of p-values using the method proposed by Holm

Usage

```
holm_adjusted_p(p = NULL)
```

Arguments

`p`	a numeric vector of p-values

Details

See the following reference: Holm 1979 https://www.jstor.org/stable/4615733 Manual for the 'p.adjust' function in the 'stats' package https://stat.ethz.ch/R-manual/R-devel/library/stats/html/p.adjust.html

Examples

```
holm_adjusted_p(c(.05, .01))
holm_adjusted_p(c(.05, .05, .05))
```

`identical_all` *Check whether all inputs are identical*

Description

Check whether all inputs are identical

Usage

```
identical_all(...)
```

Arguments

`...`	two or more R objects. If a vector or list is entered as an input, the function will test whether the vector's or list's elements are identical.

Value

the output will be `TRUE` if all inputs are identical or `FALSE` if not

Examples

```
identical_all(1:3, 1:3) # should return TRUE
identical_all(1:3, 1:3, 1:3, 1:3, 1:3) # should return TRUE
identical_all(1:3, 1:3, 1:3, 1:3, 1:3, 1:4) # should return FALSE
identical_all(1:10) # should return FALSE
identical_all(rep(1, 100)) # should return TRUE
identical_all(list(1, 1, 1)) # should return TRUE
identical_all(TRUE, FALSE) # should return FALSE
identical_all(FALSE, TRUE) # should return FALSE
```

`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. By default, `id_col_name = "id"`.
`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")
```

`install_all_dependencies`

Install all dependencies for all functions

Description

Install all dependencies for all functions in Package 'kim'.

Usage

```
install_all_dependencies()
```

Value

there will be no output from this function. Rather, dependencies of all functions in Package 'kim' will be installed.

Examples

```
## Not run:
install_all_dependencies()

## End(Not run)
```

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)
kim::kurtosis(c(1, 2, 3, 4, 5, 10))
# calculate the biased estimator (e.g., kurtosis value that
# R Package 'moments' will produce)
kim::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))
```

lenu *lenu: Length of unique values*

Description

Extract unique elements and get the length of those elements

Usage

```
lenu(x = NULL)
```

Arguments

`x`	a vector or a data frame or an array or NULL.

Value

a vector, data frame, or array-like 'x' but with duplicate elements/rows removed.

Examples

```
unique(c(10, 3, 7, 10))
lenu(c(10, 3, 7, 10))
unique(c(10, 3, 7, 10, NA))
lenu(c(10, 3, 7, 10, NA))
lenu(c("b", "z", "b", "a", NA, NA, NA))
```

`levene_test` *Levene's test*

Description

Conduct Levene's test (i.e., test the null hypothesis that the variances in different gorups are equal)

Usage

```
levene_test(
  data = NULL,
  dv_name = NULL,
  iv_1_name = NULL,
  iv_2_name = NULL,
  round_f = 2,
  round_p = 3,
  output_type = "text"
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`dv_name`	name of the dependent variable
`iv_1_name`	name of the first independent variable
`iv_2_name`	name of the second independent variable
`round_f`	number of decimal places to which to round the F-statistic from Levene's test (default = 2)
`round_p`	number of decimal places to which to round the p-value from Levene's test (default = 3)
`output_type`	If `output_type = "text"`, the output will be the results of Levene's test in a text format (i.e., character). If `output_type = "list"`, the output will be the results of Levene's test in a list format (e.g., p value, F stat, etc. as a list). By default, `output_type = "text"`

Value

the output of the function depends on the input for `output_type`. By default, the output will be the results of Levene's test in a text format (i.e., character).

Examples

```
## Not run:
levene_test(
data = mtcars, dv_name = "mpg",
iv_1_name = "vs", iv_2_name = "am")

## End(Not run)
```

`logistic_regression` *Logistic regression*

Description

Conduct a logistic regression analysis

Usage

```
logistic_regression(
  data = NULL,
  formula = NULL,
  formula_1 = NULL,
  formula_2 = NULL,
  z_values_keep = FALSE,
  constant_row_clean = TRUE,
  odds_ratio_cols_combine = TRUE,
  round_b_and_se = 3,
  round_z = 3,
  round_p = 3,
  round_odds_ratio = 3,
  round_r_sq = 3,
  round_model_chi_sq = 3,
  pretty_round_p_value = TRUE,
  print_glm_default_summary = FALSE,
  print_summary_dt_list = TRUE,
  print_model_comparison = TRUE,
  output_type = "summary_dt_list"
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`formula`	formula for estimating a single logistic regression model

`formula_1`	formula for estimating logistic regression model 1 of 2
`formula_2`	formula for estimating logistic regression model 2 of 2
`z_values_keep`	logical. Should the z values be kept in the table? (default = FALSE)
`constant_row_clean`	logical. Should the row for the constant be cleared except for b and standard error of b? (default = TRUE)
`odds_ratio_cols_combine`	logical. Should the odds ratio columns be combined? (default = TRUE)
`round_b_and_se`	number of decimal places to which to round b and standard error of b (default = 3)
`round_z`	number of decimal places to which to round z values (default = 3)
`round_p`	number of decimal places to which to round p-values (default = 3)
`round_odds_ratio`	number of decimal places to which to round odds ratios (default = 3)
`round_r_sq`	number of decimal places to which to round R-squared values (default = 3)
`round_model_chi_sq`	number of decimal places to which to round model chi-squared values (default = 3)
`pretty_round_p_value`	logical. Should the p-values be rounded in a pretty format (i.e., lower threshold: "<.001"). By default, `pretty_round_p_value = TRUE`.
`print_glm_default_summary`	logical. Should the default summary output of the glm objects be printed? (default = FALSE)
`print_summary_dt_list`	logical. Should the summaries of logistic regressions in a data table format be printed? (default = TRUE)
`print_model_comparison`	logical. Should the comparison of two logistic regression models be printed? (default = TRUE)
`output_type`	If `output_type = "summary_dt_list"` (default), the output of the function will be summaries of the two logistic regressions in a data.table format. If `output_type = "glm_object_list"`, the output of the function will be the two `glm` objects estimating logistic regression models. If `output_type = "glm_default_summary_list"`, the output of the function will be the R's default `summary` output for the two `glm` objects estimating logistic regression models. If `output_type = "model_comparison_stats"`, the output of the function will be statistics from comparison of the two logistic regression models. If `output_type = "all"`, the output of the function will be a list of the aforementioned outputs.

Value

the output will be a summary of logistic regression results, unless set otherwise by the `output_type` argument to the function.

Examples

```
logistic_regression(data = mtcars, formula = am ~ mpg)
logistic_regression(
data = mtcars,
formula_1 = am ~ mpg,
formula_2 = am ~ mpg + wt)
```

---

logistic_regression_table
Logistic regression table

Description

Construct a table of logistic regression results from the given glm object estimating a logistic regression model.

Usage

```
logistic_regression_table(
  logistic_reg_glm_object = NULL,
  z_values_keep = FALSE,
  constant_row_clean = TRUE,
  odds_ratio_cols_combine = TRUE,
  round_b_and_se = 3,
  round_z = 3,
  round_p = 3,
  round_odds_ratio = 3,
  round_r_sq = 3,
  round_model_chi_sq = 3,
  pretty_round_p_value = TRUE
)
```

Arguments

`logistic_reg_glm_object`	a glm object estimating a logistic regression model
`z_values_keep`	logical. Should the z values be kept in the table? (default = FALSE)
`constant_row_clean`	logical. Should the row for the constant be cleared except for b and standard error of b? (default = TRUE)
`odds_ratio_cols_combine`	logical. Should the odds ratio columns be combined? (default = TRUE)
`round_b_and_se`	number of decimal places to which to round b and standard error of b (default = 3)
`round_z`	number of decimal places to which to round z values (default = 3)

`round_p`	number of decimal places to which to round p-values (default = 3)
`round_odds_ratio`	number of decimal places to which to round odds ratios (default = 3)
`round_r_sq`	number of decimal places to which to round R-squared values (default = 3)
`round_model_chi_sq`	number of decimal places to which to round model chi-squared values (default = 3)
`pretty_round_p_value`	logical. Should the p-values be rounded in a pretty format (i.e., lower threshold: "<.001"). By default, `pretty_round_p_value = TRUE`.

Value

the output will be a summary of logistic regression results.

Examples

```
logistic_regression_table(logistic_reg_glm_object =
glm(formula = am ~ mpg, family = binomial(), data = mtcars))
logistic_regression_table(logistic_reg_glm_object =
glm(formula = am ~ mpg, family = binomial(), data = mtcars),
z_values_keep = TRUE, constant_row_clean = FALSE,
odds_ratio_cols_combine = FALSE)
```

`logistic_reg_w_interaction`
Logistic regression with an interaction term

Description

Conduct logistic regression for a model with an interaction between two predictor variables

Usage

```
logistic_reg_w_interaction(
  data = NULL,
  dv_name = NULL,
  iv_1_name = NULL,
  iv_2_name = NULL,
  round_p = 3,
  round_chi_sq = 2,
  dv_ordered_levels = NULL,
  iv_1_ordered_levels = NULL,
  iv_2_ordered_levels = NULL,
  one_line_summary_only = FALSE,
  p_value_interaction_only = FALSE,
  return_dt_w_binary = FALSE
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`dv_name`	name of the dependent variable (must be a binary variable)
`iv_1_name`	name of the first independent variable
`iv_2_name`	name of the second independent variable
`round_p`	number of decimal places to which to round p-values (default = 3)
`round_chi_sq`	number of decimal places to which to round chi square statistics (default = 2)
`dv_ordered_levels`	a vector with the ordered levels of the dependent variable, the first and second elements of which will be coded as 0 and 1, respectively, to run logistic regression. E.g., `dv_ordered_levels = c("fail", "pass")`
`iv_1_ordered_levels`	(only if the first independent variable is a binary variable) a vector with the ordered levels of the first independent variable, the first and second elements of which will be coded as 0 and 1, respectively, to run logistic regression. E.g., `iv_1_ordered_levels = c("control", "treatment")`
`iv_2_ordered_levels`	(only if the second independent variable is a binary variable) a vector with the ordered levels of the first independent variable, the first and second elements of which will be coded as 0 and 1, respectively, to run logistic regression. E.g., `iv_2_ordered_levels = c("male", "female")`
`one_line_summary_only`	logical. Should the output simply be a printout of a one-line summary on the interaction term? (default = FALSE)
`p_value_interaction_only`	logical. Should the output simply be a p-value of the interaction term in the logistic regression model? (default = FALSE)
`return_dt_w_binary`	logical. If `return_dt_w_binary = TRUE`, the function will return a data.table with binary variables coded as 0 or 1 (default = FALSE)

Value

the output will be a summary of logistic regression results, unless set otherwise by arguments to the function.

Examples

```
logistic_reg_w_interaction(data = mtcars, dv_name = "vs",
iv_1_name = "mpg", iv_2_name = "am")
```

loglinear_analysis *Loglinear analysis*

Description

Conduct a loglinear analysis

Usage

```
loglinear_analysis(
  data = NULL,
  dv_name = NULL,
  iv_1_name = NULL,
  iv_2_name = NULL,
  iv_1_values = NULL,
  iv_2_values = NULL,
  output = "all",
  round_p = 3,
  round_chi_sq = 2,
  mosaic_plot = TRUE,
  report_as_field = FALSE
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`dv_name`	name of the dependent variable
`iv_1_name`	name of the first independent variable
`iv_2_name`	name of the second independent variable
`iv_1_values`	restrict all analyses to observations having these values for the first independent variable
`iv_2_values`	restrict all analyses to observations having these values for the second independent variable
`output`	type of the output. If `output_type = "all"`, the function will return a results summary and print a mosaic plot. (default = "all")
`round_p`	number of decimal places to which to round p-values (default = 3)
`round_chi_sq`	number of decimal places to which to round chi-squared test statistics (default = 2)
`mosaic_plot`	If `mosaic_plot = TRUE`, a mosaic plot will be printed (default = TRUE)
`report_as_field`	If `report_as_field = TRUE`, reports summary will follow the format suggested by Andy Field (2012) (ISBN: 978-1-4462-0045-2, p. 851)

Examples

```
loglinear_analysis(data = data.frame(Titanic), "Survived", "Sex", "Age")
```

---

log_odds_ratio *Log odds ratio*

Description

Calculate log odds ratio (i.e., ln of odds ratio), as illustrated in Borenstein et al. (2009, p. 36, ISBN: 978-0-470-05724-7)

Usage

```
log_odds_ratio(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  contingency_table = NULL,
  ci = 0.95,
  var_include = FALSE,
  invert = 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 (binary outcome)
`contingency_table`	a contingency table, which can be directly entered as an input for calculating the odds ratio
`ci`	width of the confidence interval. Input can be any value less than 1 and greater than or equal to 0. By default, `ci = 0.95`. If `ci = TRUE`, the default value of 0.95 will be used. If `ci = FALSE`, no confidence interval will be estimated.
`var_include`	logical. Should the output include variance of the log of odds ratio? (default = FALSE)
`invert`	logical. Whether the inverse of the odds ratio (i.e., 1 / odds ratio) should be returned.

Examples

```
## Not run:
log_odds_ratio(data = mtcars, iv_name = "vs", dv_name = "am")
log_odds_ratio(contingency_table = matrix(c(5, 10, 95, 90), nrow = 2))
log_odds_ratio(contingency_table = matrix(c(5, 10, 95, 90), nrow = 2),
invert = TRUE)
log_odds_ratio(contingency_table = matrix(c(34, 39, 16, 11), nrow = 2))
log_odds_ratio(contingency_table = matrix(c(34, 39, 16, 11), nrow = 2),
var_include = TRUE)

## End(Not run)
```

`log_odds_ratio_to_d` *Convert log odds ratio to Cohen's d*

Description

Convert log odds ratio to Cohen'd (standardized mean difference), as illustrated in Borenstein et al. (2009, p. 47, ISBN: 978-0-470-05724-7)

Usage

```
log_odds_ratio_to_d(log_odds_ratio = NULL, unname = TRUE)
```

Arguments

`log_odds_ratio`	log odds ratio (the input can be a vector of values), which will be converted to Cohen's d
`unname`	logical. Should the names from the input be removed? (default = TRUE)

Examples

```
## Not run:
log_odds_ratio_to_d(log(1))
log_odds_ratio_to_d(log(2))

## End(Not run)
```

mad_remove_outliers *Remove outliers using the MAD method*

Description

Detect outliers in a numeric vector using the Median Absolute Deviation (MAD) method and remove or convert them. For more information on MAD, see Leys et al. (2013) doi:10.1016/j.jesp.2013.03.013

Usage

```
mad_remove_outliers(
  x = NULL,
  threshold = 2.5,
  constant = 1.4826,
  convert_outliers_to = NA,
  output_type = "converted_vector"
)
```

Arguments

`x`	a numeric vector
`threshold`	the threshold value for determining outliers. If `threshold == 2.5`, the median plus or minus 2.5 times the MAD will be the cutoff values for determining outliers. In other words, values less than the median minus 2.5 times the MAD and values greater than the median plus 2.5 times the MAD will be considered outliers. By default, `threshold == 2.5`
`constant`	scale factor for the 'mad' function in the 'stats' package. It is the constant linked to the assumed distribution. In case of normality, constant = 1.4826. By default, `constant == 1.4826`.
`convert_outliers_to`	the value to which outliers will be converted. For example, if `convert_outliers_to = NA`, the outlier values will be converted to NA values. If `convert_outliers_to = 1000`, the outlier values will be converted to 1000. By default, `convert_outliers_to == NA`.
`output_type`	type of the output. If `output_type = "converted_vector"`, the function's output will be a vector with outliers converted to the value set by the argument `convert_outliers_to`. If `output_type = "outliers"`, the function's output will be outliers in the original vector as determined by the MAD method. If `output_type = "cutoff_values"`, the function's output will be the cutoff values for determining outliers. For example, if outliers will be values less than 0 and greater than 10, the cutoff values will be 0 and 10. If `output_type = "non_outlier_values"`, the function's output will be a vector consisting only of the values that are not outliers; here, the outliers will be removed from the vector, rather than being converted to NA values. By default, `output_type = "converted_vector"`.

Examples

```
## Not run:
mad_remove_outliers(x = c(1, 3, 3, 6, 8, 10, 10, 1000))
mad_remove_outliers(x = c(1, 3, 3, 6, 8, 10, 10, 1000, -10000))
# return the vector with the outlier converted to NA values
mad_remove_outliers(
x = c(1, 3, 3, 6, 8, 10, 10, 1000, -10000),
output_type = "converted_vector")
# return the cutoff values for determining outliers
mad_remove_outliers(
x = c(1, 3, 3, 6, 8, 10, 10, 1000, -10000),
output_type = "cutoff_values")
# return the outliers
mad_remove_outliers(
x = c(1, 3, 3, 6, 8, 10, 10, 1000, -10000),
output_type = "outliers")
mad_remove_outliers(
x = c(1, 3, 3, 6, 8, 10, 10, 1000, -10000),
output_type = "non_outlier_values")

## End(Not run)
```

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,
  iv_level_order = 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)
iv_level_order	order of levels in the independent variable. By default, it will be set as levels of the independent variable ordered using R's base function `sort`.
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")
```

`matrix_prep_dt` *Prepare a two-column data.table that will be used to fill values in a matrix*

Description

Prepare a two-column data.table that will be used to fill values in a matrix

Usage

```
matrix_prep_dt(row_var_names = NULL, col_var_names = NULL)
```

Arguments

`row_var_names`	a vector of variable names, each of which will be header of a row in the eventual matrix
`col_var_names`	a vector of variable names, each of which will be header of a column in the eventual matrix

Examples

```
matrix_prep_dt(
  row_var_names = c("mpg", "cyl"),
  col_var_names = c("hp", "gear")
)
```

`mean_center` *Mean center*

Description

Mean-center a variable, i.e., subtract the mean of a numeric vector from each value in the numeric vector

Usage

```
mean_center(x)
```

Arguments

`x`	a numeric vector; though not thoroughly tested, the function can accept a matrix as an input.

Examples

```
mean_center(1:5)
mean_center(1:6)
# if the input is a matrix
matrix(1:9, nrow = 3)
mean_center(matrix(1:9, nrow = 3))
```

`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.

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)

Details

This function requires installing Package 'mediation' v4.5.0 (or possibly a higher version) by Tingley et al. (2019), and uses the source code from a function in the package. https://cran.r-project.org/package=mediation

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(s) of the column(s) that will contain the ID values in the two data tables. The name(s) of the ID column(s) 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 column names were duplicated. In cases where only one column was used as the 'id' column (which is the most common case), `silent = FALSE` will also print messages regarding how many input ID values were duplicated. By default, `silent = FALSE`.

Value

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

Examples

```
## Example 1: Typical Usage
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, 99, 4),
e = 6:8,
b = c("p", "q", "r"),
d = c(TRUE, FALSE, FALSE))
# check the two example data tables
```

```
data_1
data_2
# check the result of merging the two data tables above and
# note how data_1 (the upper left portion) is intact in the resulting
# data table
merge_data_tables(dt1 = data_1, dt2 = data_2, id = "id_col")
# compare the result with above with the result from the `merge` function
merge(data_1, data_2, by = "id_col", all = TRUE)
## Example 2: Some values can be converted
data_3 <- data.table::data.table(
id_col = 99,
a = "abc",
b = TRUE,
c = TRUE)
data_1
data_3
merge_data_tables(data_1, data_3, id = "id_col")
# In the example above, note how the value of TRUE gets
# converted to 1 in the last row of Column 'b' in the resulting data table
## Example 3: A simpler case
data_4 <- data.table::data.table(
id_col = c(5, 3),
a = c("a", NA))
data_5 <- data.table::data.table(
id_col = 1,
a = 2)
# check the two example data tables
data_4
data_5
merge_data_tables(data_4, data_5, id = "id_col")
## Example 4: Merging data tables using multiple ID columns
data_6 <- data.table::data.table(
id_col_1 = 3:1,
id_col_2 = c("a", "b", "c"),
id_col_3 = 4:6,
a = 7:9,
b = 10:12)
data_7 <- data.table::data.table(
id_col_1 = c(3, 2),
id_col_3 = c(3, 5),
id_col_2 = c("a", "b"),
c = 13:14,
a = 15:16)
# check the example data sets
data_6
data_7
# merge data sets using the three id columns
suppressWarnings(merge_data_tables(
dt1 = data_6,
dt2 = data_7,
id = c("id_col_1", "id_col_2", "id_col_3")))
```

`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(s) of the column(s) that will contain the ID values in the two data tables. The name(s) of the ID column(s) 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 column names were duplicated. In cases where only one column was used as the 'id' column (which is the most common case), `silent = FALSE` will also print messages regarding how many input ID values were duplicated. By default, `silent = 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")
```

---

mixed_anova_2_way *Mixed ANOVA 2-Way (Two-Way Mixed ANOVA)*

Description

Conduct a two-way mixed analysis of variance (ANOVA).

Usage

```
mixed_anova_2_way(
  data = NULL,
  iv_name_bw_group = NULL,
  repeated_measures_col_names = NULL,
  iv_name_bw_group_values = NULL,
  colors = NULL,
  error_bar = "ci",
  position_dodge = 0.13,
  legend_title = NULL,
  x_axis_expansion_add = c(0.2, 0.03),
  x_axis_title = NULL,
  y_axis_title = "Mean",
  output = "all"
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name_bw_group`	name of the between-group independent variable
`repeated_measures_col_names`	names of the columns containing the repeated measures
`iv_name_bw_group_values`	restrict all analyses to observations having these values for the between-group independent variable
`colors`	colors of the dots and lines connecting means (default = NULL) If there are exactly two repeated measures, then, by default, `colors = c("red", "blue")`
`error_bar`	if `error_bar = "ci"` error bars will be a 95% confidence interval; if `error_bar = "se"`, error bars will be +/-1 standard error. By default, `error_bar = "ci"`

`position_dodge`	by how much should the group means and error bars be horizontally offset from each other so as not to overlap? (default = 0.13)
`legend_title`	a character for the legend title. If no input is entered, then, by default, the legend title will be removed.
`x_axis_expansion_add`	inputs for the `add` parameter of the `expand` argument. The first and second values respectively determine the amount of space to add to the left and right along the x-axis. By default, `x_axis_expansion_add = c(0.2, 0.03)` which means that space with the width of 0.2 will be added to the left, and space with the width of 0.03 will be added to the right.
`x_axis_title`	a character string for the x-axis title. If `x_axis_title == FALSE`, which is the default value, the x-axis title will be removed.
`y_axis_title`	a character string for the y-axis title (default = "Mean"). If `x_axis_title == FALSE`, the y-axis title will be removed.
`output`	output type can be one of the following: `"plot"`, `"all"`

Details

The following package(s) must be installed prior to running this function: Package 'afex' v3.0.9 (or possibly a higher version) by Fox et al. (2020), https://cran.r-project.org/package=car

Examples

```
mixed_anova_2_way(
  data = iris, iv_name_bw_group = "Species",
  repeated_measures_col_names = c("Sepal.Length", "Petal.Length"))
g1 <- mixed_anova_2_way(
  data = iris, iv_name_bw_group = "Species",
  repeated_measures_col_names = c("Sepal.Length", "Petal.Length"),
  error_bar = "se",
  output = "plot")
```

`modes_of_objects` *Find modes of objects*

Description

Find modes of objects

Usage

```
modes_of_objects(...)
```

Arguments

`...`	R objects.

Value

the output will be a data.table listing objects and their mods.

Examples

```
modes_of_objects(
TRUE, FALSE, 1L, 1:3, 1.1, c(1.2, 1.3), "abc", 1 + 2i, intToBits(1L))
```

---

multiple_regression *Multiple regression*

Description

Conduct multiple regression analysis and summarize the results in a data.table.

Usage

```
multiple_regression(
  data = NULL,
  formula = NULL,
  vars_to_mean_center = NULL,
  mean_center_vars = NULL,
  sigfigs = NULL,
  round_digits_after_decimal = NULL,
  round_p = NULL,
  pretty_round_p_value = TRUE,
  return_table_upper_half = FALSE,
  round_r_squared = 3,
  round_f_stat = 2,
  prettify_reg_table_col_names = TRUE,
  silent = FALSE,
  save_as_png = FALSE,
  png_name = NULL,
  width = 1600,
  height = 1200,
  units = "px",
  res = 200
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`formula`	a formula object for the regression equation
`vars_to_mean_center`	(deprecated) a character vector specifying names of variables that will be mean-centered before the regression model is estimated

`mean_center_vars`	a character vector specifying names of variables that will be mean-centered before the regression model is estimated
`sigfigs`	number of significant digits to round to
`round_digits_after_decimal`	round to nth digit after decimal (alternative to `sigfigs`)
`round_p`	number of decimal places to round p values (overrides all other rounding arguments)
`pretty_round_p_value`	logical. Should the p-values be rounded in a pretty format (i.e., lower threshold: "<.001"). By default, `pretty_round_p_value = TRUE`.
`return_table_upper_half`	logical. Should only the upper part of the table be returned? By default, `return_table_upper_half = FALSE`.
`round_r_squared`	number of digits after the decimal both r-squared and adjusted r-squared values should be rounded to (default 3)
`round_f_stat`	number of digits after the decimal the f statistic of the regression model should be rounded to (default 2)
`prettify_reg_table_col_names`	logical. Should the column names of the regression table be made pretty (e.g., change "std_beta" to "Std. Beta")? (Default = `TRUE`)
`silent`	If `silent = FALSE`, a message regarding mean-centered variables will be printed. If `silent = TRUE`, this message will be suppressed. By default, `silent = FALSE`.
`save_as_png`	if `save_as_png = TRUE`, the regression table will be saved as a PNG file (default = FALSE).
`png_name`	name of the PNG file to be saved. By default, the name will be "mult_reg_" followed by a timestamp of the current time. The timestamp will be in the format, jan_01_2021_1300_10_000001, where "jan_01_2021" would indicate January 01, 2021; 1300 would indicate 13:00 (i.e., 1 PM); and 10_000001 would indicate 10.000001 seconds after the hour.
`width`	width of the PNG file (default = 1600)
`height`	height of the PNG file (default = 1200)
`units`	the units for the `width` and `height` arguments. Can be `"px"` (pixels), `"in"` (inches), `"cm"`, or `"mm"`. By default, `units = "px"`.
`res`	The nominal resolution in ppi which will be recorded in the png file, if a positive integer. Used for units other than the default. By default, `res = 200`

Details

To include standardized beta(s) in the regression results table, the following package(s) must be installed prior to running the function: Package 'lm.beta' v1.5-1 (or possibly a higher version) by Stefan Behrendt (2014), https://cran.r-project.org/package=lm.beta

Value

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

Examples

```
multiple_regression(data = mtcars, formula = mpg ~ gear * cyl)
multiple_regression(
data = mtcars, formula = mpg ~ gear * cyl,
mean_center_vars = "gear",
round_digits_after_decimal = 2)
multiple_regression(
data = mtcars, formula = mpg ~ gear * cyl,
png_name = "mtcars reg table 1")
```

---

noncentrality_parameter

Find noncentrality parameter

Description

Find noncentrality parameter

Usage

```
noncentrality_parameter(t_stat, df, initial_value = 0, ci = 0.95)
```

Arguments

`t_stat`	the t-statistic associated with the noncentrality parameters
`df`	degrees of freedom associated with the noncentrality parameters
`initial_value`	initial value of the noncentrality parameter for optimization (default = 0). Adjust this value if results look strange.
`ci`	width of the confidence interval associated with the noncentrality parameters (default = 0.95)

Examples

```
noncentrality_parameter(4.29, 9)
```

`odds_ratio`	*Odds ratio*

Description

Calculate odds ratio, as illustrated in Borenstein et al. (2009, pp. 33-36, ISBN: 978-0-470-05724-7)

Usage

```
odds_ratio(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  contingency_table = NULL,
  ci = 0.95,
  round_ci_limits = 2,
  invert = 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 (binary outcome)
`contingency_table`	a contingency table, which can be directly entered as an input for calculating the odds ratio
`ci`	width of the confidence interval. Input can be any value less than 1 and greater than or equal to 0. By default, `ci = 0.95`. If `ci = TRUE`, the default value of 0.95 will be used. If `ci = FALSE`, no confidence interval will be estimated.
`round_ci_limits`	number of decimal places to which to round the limits of the confidence interval (default = 2)
`invert`	logical. Whether the inverse of the odds ratio (i.e., 1 / odds ratio) should be returned.

Examples

```
## Not run:
odds_ratio(data = mtcars, iv_name = "vs", dv_name = "am")
odds_ratio(data = mtcars, iv_name = "vs", dv_name = "am", ci = 0.9)
odds_ratio(contingency_table = matrix(c(5, 10, 95, 90), nrow = 2))
odds_ratio(contingency_table = matrix(c(5, 10, 95, 90), nrow = 2),
invert = TRUE)
odds_ratio(contingency_table = matrix(c(34, 39, 16, 11), nrow = 2))

## End(Not run)
```

`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))
```

`outlier` *Outlier*

Description

Return outliers in a vector

Usage

```
outlier(x = NULL, iqr = 1.5, na.rm = TRUE, type = 7, unique_outliers = FALSE)
```

Arguments

`x`	a numeric vector
`iqr`	a nonnegative constant by which interquartile range (IQR) will be multiplied to build a "fence," outside which observations will be considered outliers. For example, if `iqr = 1.5`, IQR * 1.5 will be the "fence" outside which observations will be considered to be outliers. By default, `iqr = 1.5`.
`na.rm`	logical. `na.rm` argument to be passed onto the 'quantile' function in the 'stats' package. If true, any NA and NaN's are removed from x before the quantiles are computed.
`type`	`type` argument to be passed onto the 'quantile' function in the 'stats' package. An integer between 1 and 9 selecting one of the nine quantile algorithms detailed below to be used. Type '?stats::quantile' for details. By default, `type = 7`
`unique_outliers`	logical. If `unique_outliers = TRUE`, the function will return the unique outlier values. If `unique_outliers = FALSE`, the function will return all the outlier values in the vector `x`. By default, `unique_outliers = FALSE`.

Value

the output will be a numeric vector with outliers removed.

Examples

```
# Example 1
outlier(c(1:10, 100))
# The steps below show how the outlier, 100, was obtained
# v1 is the vector of interest
v1 <- c(1:10, 100)
# quantile
stats::quantile(v1)
# first and third quartiles
q1 <- stats::quantile(v1, 0.25)
q3 <- stats::quantile(v1, 0.75)
# interquartile range
interquartile_range <- unname(q3 - q1)
# fence, using the default 1.5 as the factor to multiply the IQR
cutoff_low <- unname(q1 - 1.5 * interquartile_range)
cutoff_high <- unname(q3 + 1.5 * interquartile_range)
v1[v1 < cutoff_low | v1 > cutoff_high]
```

`overlapping_interval` *Find the overlapping interval of two ranges.*

Description

This function should be applied to cases where the two ranges are inclusive of both endpoints. For example, the function can work for a pair of ranges like [0, 1] and [3, 4] but not for pairs like [0, 1\) and \(3, 5\)

Usage

```
overlapping_interval(
  interval_1_begin = NULL,
  interval_1_end = NULL,
  interval_2_begin = NULL,
  interval_2_end = NULL
)
```

Arguments

`interval_1_begin`	a number at which the first interval begins (the left INCLUSIVE endpoint of interval 1)
`interval_1_end`	a number at which the first interval ends (the right INCLUSIVE endpoint of interval 1)
`interval_2_begin`	a number at which the second interval begins (the left INCLUSIVE endpoint of interval 2)
`interval_2_end`	a number at which the second interval ends (the right INCLUSIVE endpoint of interval 2)

Value

the output will be `NULL` if there is no overlapping region or a vector of the endpoints of the overlapping interval.

Examples

```
overlapping_interval(1, 3, 2, 4)
overlapping_interval(1, 2.22, 2.22, 3)
```

---

p0 *Paste0*

Description

A shorthand for the function `paste0` Concatenate vectors after converting to character.

Usage

```
p0(..., collapse = NULL, recycle0 = FALSE)
```

Arguments

`...`	one or more R objects, to be converted to character vectors. This is the same argument that would be used in the `paste0` function.
`collapse`	an optional character string to separate the results. Not NA_character_. This is the same argument that would be used in the `paste0` function.
`recycle0`	logical indicating if zero-length character arguments should lead to the zero-length character(0) after the sep-phase (which turns into "" in the collapse-phase, i.e., when collapse is not NULL). This is the same argument that would be used in the `paste0` function.

Examples

```
paste0("a", "b")
p0("a", "b")
```

`package_list_default` *Packages - List the default packages*

Description

List the default packages in R

Usage

```
package_list_default(package_type = c("base", "recommended"))
```

Arguments

`package_type`	a vector of package types. By default, `package_type = c("base", "recommended")`

Examples

```
package_list_default()
package_list_default(package_type = "base")
```

parallel_analysis *Parallel analysis*

Description

Conducts a parallel analysis to determine how many factors to retain in a factor analysis.

Usage

```
parallel_analysis(
  data = NULL,
  names_of_vars = NULL,
  iterations = NULL,
  percentile_for_eigenvalue = 95,
  line_types = c("dashed", "solid"),
  colors = c("red", "blue"),
  eigenvalue_random_label_x_pos = NULL,
  eigenvalue_random_label_y_pos = NULL,
  unadj_eigenvalue_label_x_pos = NULL,
  unadj_eigenvalue_label_y_pos = NULL,
  label_offset_percent = 2,
  label_size = 6,
  dot_size = 5,
  line_thickness = 1.5,
  y_axis_title_vjust = 0.8,
  title_text_size = 26,
  axis_text_size = 22
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`names_of_vars`	names of the variables
`iterations`	number of random data sets. If no input is entered, this value will be set as 30 * number of variables.
`percentile_for_eigenvalue`	percentile used in estimating bias (default = 95).
`line_types`	types of the lines connecting eigenvalues. By default, `line_types = c("dashed", "solid")`
`colors`	size of the dots denoting eigenvalues (default = 5).
`eigenvalue_random_label_x_pos`	(optional) x coordinate of the label for eigenvalues from randomly generated data.
`eigenvalue_random_label_y_pos`	(optional) y coordinate of the label for eigenvalues from randomly generated data.

`unadj_eigenvalue_label_x_pos`	(optional) x coordinate of the label for unadjusted eigenvalues
`unadj_eigenvalue_label_y_pos`	(optional) y coordinate of the label for unadjusted eigenvalues
`label_offset_percent`	How much should labels for the eigenvalue curves be offset, as a percentage of the plot's x and y range? (default = 2)
`label_size`	size of the labels for the eigenvalue curves (default = 6).
`dot_size`	size of the dots denoting eigenvalues (default = 5).
`line_thickness`	thickness of the eigenvalue curves (default = 1.5).
`y_axis_title_vjust`	position of the y axis title as a proportion of the range (default = 0.8).
`title_text_size`	size of the plot title (default = 26).
`axis_text_size`	size of the text on the axes (default = 22).

Details

The following package(s) must be installed prior to running the function: Package 'paran' v1.5.2 (or possibly a higher version) by Alexis Dinno (2018), `https://cran.r-project.org/package=paran`

Examples

```
parallel_analysis(
  data = mtcars, names_of_vars = c("disp", "hp", "drat"))
# parallel_analysis(
# data = mtcars, names_of_vars = c("carb", "vs", "gear", "am"))
```

`percentile_rank` *Percentile rank*

Description

Calculate percentile rank of each value in a vector

Usage

```
percentile_rank(vector)
```

Arguments

`vector`	a numeric vector

Examples

```
percentile_rank(1:5)
percentile_rank(1:10)
percentile_rank(1:100)
```

---

pivot_table *Pivot Table*

Description

Create a pivot table.

Usage

```
pivot_table(
  data = NULL,
  row_names = NULL,
  col_names = NULL,
  function_as_character = NULL,
  sigfigs = 3,
  output = "dt",
  remove_col_names = TRUE
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`row_names`	names of variables for constructing rows
`col_names`	names of variables for constructing columns independent variables
`function_as_character`	function to perform for each cell in the pivot table
`sigfigs`	number of significant digits to which to round values in the pivot table (default = 3)
`output`	type of output. If `output = "dt"`, the function's output will be a pivot table in a data.table format. If `output = "subsets"`, the function's output will be a list of data tables that are subsets representing each cell in the pivot table. By default, `output = "dt"`
`remove_col_names`	logical. Should the column names (i.e., v1, v2, ...) be removed in the data table output?

Value

the output will be a contingency table in a data.table format

Examples

```
pivot_table(
  data = mtcars, col_names = "am", row_names = c("cyl", "vs"),
  function_as_character = "mean(mpg)")
pivot_table(
  data = mtcars, col_names = "am", row_names = c("cyl", "vs"),
  function_as_character = "sum(mpg < 17)")
pivot_table(
  data = mtcars, col_names = "am", row_names = c("cyl", "vs"),
  function_as_character =
  "round(sum(mpg < 17) / sum(!is.na(mpg)) * 100, 0)")
```

`plot_group_means` *Plot group means*

Description

Creates a plot of sample means and error bars by group.

Usage

```
plot_group_means(
  data = NULL,
  dv_name = NULL,
  iv_name = NULL,
  na.rm = TRUE,
  error_bar = "ci",
  error_bar_range = 0.95,
  error_bar_tip_width = 0.13,
  error_bar_thickness = 1,
  error_bar_caption = TRUE,
  lines_connecting_means = TRUE,
  line_colors = NULL,
  line_types = NULL,
  line_thickness = 1,
  line_size = NULL,
  dot_size = 3,
  position_dodge = 0.13,
  x_axis_title = NULL,
  y_axis_title = NULL,
  y_axis_title_vjust = 0.85,
  legend_title = NULL,
  legend_position = "right"
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`dv_name`	name of the dependent variable
`iv_name`	name(s) of the independent variable(s). Up to two independent variables can be supplied.
`na.rm`	logical. If `na.rm = TRUE`, NA values in independent and dependent variables will be removed before calculating group means.
`error_bar`	if `error_bar = "se"`; error bars will be +/-1 standard error, if `error_bar = "ci"` error bars will be a confidence interval; if `error_bar = "pi"`, error bars will be a prediction interval
`error_bar_range`	width of the confidence or prediction interval (default = 0.95 for 95 percent confidence or prediction interval). This argument will not apply when `error_bar = "se"`
`error_bar_tip_width`	graphically, width of the segments at the end of error bars (default = 0.13)
`error_bar_thickness`	thickness of the error bars (default = 1)
`error_bar_caption`	should a caption be included to indicate the width of the error bars? (default = TRUE).
`lines_connecting_means`	logical. Should lines connecting means within each group be drawn? (default = TRUE)
`line_colors`	colors of the lines connecting means (default = NULL) If the second IV has two levels, then by default, `line_colors = c("red", "blue")`
`line_types`	types of the lines connecting means (default = NULL) If the second IV has two levels, then by default, `line_types = c("solid", "dashed")`
`line_thickness`	thickness of the lines connecting group means (default = 1)
`line_size`	Deprecated. Use the 'linewidth' argument instead. (default = 1)
`dot_size`	size of the dots indicating group means (default = 3)
`position_dodge`	by how much should the group means and error bars be horizontally offset from each other so as not to overlap? (default = 0.13)
`x_axis_title`	a character string for the x-axis title. If no input is entered, then, by default, the first value of `iv_name` will be used as the x-axis title.
`y_axis_title`	a character string for the y-axis title. If no input is entered, then, by default, `dv_name` will be used as the y-axis title.
`y_axis_title_vjust`	position of the y axis title (default = 0.85). By default, `y_axis_title_vjust = 0.85`, which means that the y axis title will be positioned at 85% of the way up from the bottom of the plot.
`legend_title`	a character for the legend title. If no input is entered, then, by default, the second value of `iv_name` will be used as the legend title. If `legend_title = FALSE`, then the legend title will be removed.

`legend_position`	position of the legend: `"none"`, `"top"`, `"right"`, `"bottom"`, `"left"`, `"none"` (default = `"right"`)

Value

by default, the output will be a ggplot object. If `output = "table"`, the output will be a data.table object.

Examples

```
plot_group_means(data = mtcars, dv_name = "mpg", iv_name = c("vs", "am"))
plot_group_means(
  data = mtcars, dv_name = "mpg", iv_name = c("vs", "am"),
  error_bar = "se"
)
plot_group_means(
  data = mtcars, dv_name = "mpg", iv_name = c("vs", "am"),
  error_bar = "pi", error_bar_range = 0.99
)
# set line colors and types manually
plot_group_means(
data = mtcars, dv_name = "mpg", iv_name = c("vs", "am"),
line_colors = c("green4", "purple"),
line_types = c("solid", "solid"))
# remove axis titles
plot_group_means(
data = mtcars, dv_name = "mpg", iv_name = c("vs", "am"),
x_axis_title = FALSE, y_axis_title = FALSE, legend_title = FALSE)
```

`pm` *Paste for message*

Description

Combines the base functions paste0 and message

Usage

```
pm(..., collapse = NULL)
```

Arguments

`...`	one or more R objects, to be converted to character vectors. Input(s) to this argument will be passed onto the paste0 function.
`collapse`	an optional character string to separate the results. Not `NA_character_`. Input(s) to this argument will be passed onto the paste0 function.

Value

there will be no output from this function. Rather, a message will be generated from the arguments.

Examples

```
pm("hello", 123)
pm(c("hello", 123), collapse = ", ")
```

`population_variance` *Population variance of a vector*

Description

Calculates the population variance, rather than the sample variance, of a vector

Usage

```
population_variance(vector, na.rm = TRUE)
```

Arguments

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

Examples

```
population_variance(1:4)
var(1:4)
```

`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

Round p-values to the desired number of decimals and remove leading 0s before the decimal.

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(0.00001)
pretty_round_p_value(0.00001, round_digits_after_decimal = 4)
pretty_round_p_value(0.00001, round_digits_after_decimal = 5)
# WARNING: the line of code below adding precision that may be unwarranted
pretty_round_p_value(0.00001, round_digits_after_decimal = 6)
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, 0.0009), include_p_equals = TRUE)
```

`pretty_round_r` *Pretty round r*

Description

Round correlation coefficients in APA style (7th Ed.)

Usage

```
pretty_round_r(r = NULL, round_digits_after_decimal = 2)
```

Arguments

`r`	a (vector of) correlation coefficient(s)
`round_digits_after_decimal`	how many digits after the decimal point should the p-value be rounded to? (default = 2)

Value

the output will be a character vector of correlation coefficient(s).

Examples

```
pretty_round_r(r = -0.123)
pretty_round_r(c(-0.12345, 0.45678), round_digits_after_decimal = 3)
pretty_round_r(c(-0.12, 0.45), round_digits_after_decimal = 4)
```

print_loop_progress *print loop progress*

Description

Print current progress inside a loop (e.g., for loop or lapply)

Usage

```
print_loop_progress(
  iteration_number = NULL,
  iteration_start = 1,
  iteration_end = NULL,
  text_before = "",
  percent = 1,
  output_method = "cat"
)
```

Arguments

`iteration_number`	current number of iteration
`iteration_start`	iteration number at which the loop begins (default = 1)
`iteration_end`	iteration number at which the loop ends.
`text_before`	text to add before "Loop Progress..." By default, it is set to be blank, i.e., `text_before = ""`
`percent`	if `percent = 1`, progress level will be printed at every 1 percent progress (default = 1)
`output_method`	if `output_method = "cat"`, progress level will be printed using the 'cat' function; if `output_method = "return"`, progress level will be returned as the output of the function (default = "cat")

Examples

```
for (i in seq_len(250)) {
  Sys.sleep(0.001)
  print_loop_progress(
    iteration_number = i,
    iteration_end = 250)
}
```

```
unlist(lapply(seq_len(7), function (i) {
  Sys.sleep(0.1)
  print_loop_progress(
    iteration_number = i,
    iteration_end = 7)
  return(i)
}))
```

---

proportion_of_values_in_vector

Proportion of given values in a vector

Description

Proportion of given values in a vector

Usage

```
proportion_of_values_in_vector(
  values = NULL,
  vector = NULL,
  na.exclude = TRUE,
  output_type = "proportion",
  silent = FALSE,
  conf.level = 0.95,
  correct_yates = TRUE
)
```

Arguments

`values`	a set of values that will count as successes (hits)
`vector`	a numeric or character vector containing successes (hits) and failures (misses)
`na.exclude`	if `TRUE`, NA values will be removed both from `vector` and `values` before calculation (default = TRUE).
`output_type`	By default, `output_type = "proportion"`. If `output_type = "proportion"`, the function will return the calculated proportion; if `output_type = "se"`, the function will return the standard error of the sample proportion; if `output_type = "dt"`, the function will return the the data table of proportion and confidence intervals.
`silent`	If `silent = TRUE`, no message will be printed regarding number of NA values or confidence interval. (default = FALSE)
`conf.level`	confidence level of the returned confidence interval. Input to this argument will be passed onto the conf.level argument in the `prop.test` function from the default stats package.
`correct_yates`	a logical indicating whether Yates' continuity correction should be applied where possible (default = TRUE). Input to this argument will be passed onto the `correct` argument in the `prop.test` function from the default stats package.

Examples

```
proportion_of_values_in_vector(
  values = 2:3, vector = c(rep(1:3, each = 10), rep(NA, 10))
)
proportion_of_values_in_vector(
  values = 2:3, vector = c(rep(1:3, each = 10), rep(NA, 10)),
  output_type = "se"
)
proportion_of_values_in_vector(
  values = 2:3, vector = c(rep(1:3, each = 10), rep(NA, 10)),
  conf.level = 0.99
)
proportion_of_values_in_vector(
  values = c(2:3, NA), vector = c(rep(1:3, each = 10), rep(NA, 10)),
  na.exclude = FALSE
)
```

`q_stat_test_homo_r` *Q statistic for testing homogeneity of correlations*

Description

Calculate the Q statistic to test for homogeneity of correlation coefficients. See p. 235 of the book Hedges & Olkin (1985), Statistical Methods for Meta-Analysis (ISBN: 0123363802).

Usage

```
q_stat_test_homo_r(z = NULL, n = NULL)
```

Arguments

`z`	a vector of z values
`n`	a vector of sample sizes which will be used to calculate the weights, which in turn will be used to calculate the weighted z.

Value

the output will be a weighted z value.

Examples

```
q_stat_test_homo_r(1:3, c(100, 200, 300))
q_stat_test_homo_r(z = c(1:3, NA), n = c(100, 200, 300, NA))
```

`read_csv` *Read a csv file*

Description

Read a csv file

Usage

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

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.
`dirname`	a character string of the directory containing the csv file, e.g., `dirname = "c:/Users/Documents"`
`...`	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)
```

`read_sole_csv` *Read the sole csv file in the working directory*

Description

Read the sole csv file in the working directory

Usage

```
read_sole_csv(head = FALSE, ...)
```

Arguments

`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

```
mydata <- read_sole_csv()
mydata <- read_sole_csv(head = TRUE)
mydata <- read_sole_csv(fill = TRUE, nrows = 5)
```

`regex_match` *Regular expression matches*

Description

Returns elements of a character vector that match the given regular expression

Usage

```
regex_match(regex = NULL, vector = NULL, silent = FALSE, perl = FALSE)
```

Arguments

`regex`	a regular expression provided, a default theme will be used.
`vector`	a character vector in which to search for regular expression matches, or a data table whose column names will be searched
`silent`	logical. If `silent = FALSE`, a report on regular expression matches will be printed. If `silent = TRUE`, the report on regular expression matches will not be printed. By default, `silent = FALSE`
`perl`	logical. Should Perl-compatible regexps be used?

Examples

```
regex_match("p$", names(mtcars))

colnames_ending_with_p <- regex_match("p$", names(mtcars))
```

rel_pos_of_value_in_vector

Find relative position of a value in a vector

Description

Find relative position of a value in a vector that may or may not contain the value

Usage

```
rel_pos_of_value_in_vector(value = NULL, vector = NULL)
```

Arguments

value	a value whose relative position is to be searched in a vector
vector	a numeric vector

Value

a number indicating the relative position of the value in the vector

Examples

```
rel_pos_of_value_in_vector(value = 3, vector = c(2, 4))
rel_pos_of_value_in_vector(value = 3, vector = c(2, 6))
rel_pos_of_value_in_vector(value = 3, vector = 1:3)
```

rel_value_of_pos_in_vector

Find relative value of a position in a vector

Description

Find relative value of a position in a vector

Usage

```
rel_value_of_pos_in_vector(vector = NULL, position = NULL)
```

Arguments

vector	a numeric vector
position	position of a vector

Value

a number indicating the relative value of the position in the vector

Examples

```
rel_value_of_pos_in_vector(vector = c(0, 100), position = 1.5)
rel_value_of_pos_in_vector(vector = 2:4, position = 2)
rel_value_of_pos_in_vector(vector = c(2, 4, 6), position = 2.5)
```

`remove_from_vector` *Remove from a vector*

Description

Remove certain values from a vector

Usage

```
remove_from_vector(values = NULL, vector = NULL, silent = FALSE)
```

Arguments

`values`	a single value or a vector of values which will be removed from the target vector
`vector`	a character or numeric vector
`silent`	if `silent = FALSE`, a summary of values removed will be printed; if `silent = TRUE`, such summary will not be printed. By default, `silent = FALSE`

Value

the output will be a vector with the given values removed.

Examples

```
remove_from_vector(values = 1, vector = 1:3)
remove_from_vector(values = NA, vector = c(1:3, NA))
remove_from_vector(values = c(1, NA), vector = c(1:3, NA))
remove_from_vector(values = 1:5, vector = 1:10)
```

`remove_user_installed_pkgs`
Remove all user installed packages

Description

Remove all user installed packages

Usage

```
remove_user_installed_pkgs(
  exceptions = NULL,
  type_of_pkg_to_keep = c("base", "recommended"),
  keep_kim = FALSE
)
```

Arguments

`exceptions`	a character vector of names of packages to keep
`type_of_pkg_to_keep`	a character vector indicating types of packages to keep. The default, `type_of_pkg_to_keep = c("base", "recommended")`, keeps all base and recommended packages that come with R when R is installed.
`keep_kim`	logical. If `keep_kim = FALSE`, Package 'kim' will be removed along with all other user-installed packages. If `keep_kim = TRUE`, Package 'kim' will not be removed. By default, `keep_kim = FALSE`

Examples

```
## Not run:
remove_user_installed_pkgs()

## End(Not run)
```

`repeated_measures_anova`
Repeated-Measures ANVOA

Description

Conduct a repeated-measures analysis of variance (ANOVA). This analysis will be appropriate for within-subjects experimental design.

Usage

```
repeated_measures_anova(
  data = NULL,
  p_col_name = NULL,
  measure_vars = NULL,
  histograms = TRUE,
  round_w = 2,
  round_epsilon = 2,
  round_df_model = 2,
  round_df_error = 2,
  round_f = 2,
  round_ges = 2
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`p_col_name`	name of the column identifying participants
`measure_vars`	names of the columns containing repeated measures (within-subjects variables)
`histograms`	logical. If `histograms = TRUE`, histograms of the repeated measures will be plotted. If `histograms = FALSE`, no histograms will be plotted.
`round_w`	number of decimal places to which to round W statistic from Mauchly's test (default = 2)
`round_epsilon`	number of decimal places to which to round the epsilon statistic from Greenhouse-Geisser or Huynh-Feldt correction (default = 2)
`round_df_model`	number of decimal places to which to round the corrected degrees of freedom for model (default = 2)
`round_df_error`	number of decimal places to which to round the corrected degrees of freedom for error (default = 2)
`round_f`	number of decimal places to which to round the F statistic (default = 2)
`round_ges`	number of decimal places to which to round generalized eta-squared (default = 2)

Details

The following package(s) must be installed prior to running the function: Package 'ez' v4.4-0 (or possibly a higher version) by Michael A Lawrence (2016), `https://cran.r-project.org/package=ez`

Examples

```
## Not run:
repeated_measures_anova(
  data = mtcars, p_col_name = "cyl", measure_vars = c("wt", "qsec"))

## End(Not run)
```

`replace_values_in_dt` *Replace values in a data table*

Description

Replace values in a data.table

Usage

```
replace_values_in_dt(
  data = NULL,
  old_values = NULL,
  new_values = NULL,
  silent = FALSE
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`old_values`	a vector of old values that need to be replaced
`new_values`	a new value or a vector of new values that will replace the old values
`silent`	If `silent = FALSE`, a message will be printed regarding how many values were replaced. If `silent = TRUE`, no message will be printed regarding how many values were replaced. (default = FALSE)

Examples

```
replace_values_in_dt(data = mtcars, old_values = 21.0, new_values = 888)
replace_values_in_dt(data = mtcars, old_values = c(0, 1), new_values = 999)
replace_values_in_dt(
data = mtcars, old_values = c(0, 1), new_values = 990:991)
replace_values_in_dt(
data = data.table::data.table(a = NA_character_, b = NA_character_),
old_values = NA, new_values = "")
```

`robust_regression` *Robust regression (bootstrapped regression)*

Description

Estimate coefficients in a multiple regression model by bootstrapping.

Usage

```
robust_regression(
  data = NULL,
  formula = NULL,
  sigfigs = NULL,
  round_digits_after_decimal = NULL,
  iterations = 1000
)
```

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`)
`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.

Details

The following package(s) must be installed prior to running this function: Package 'boot' v1.3-26 (or possibly a higher version) by Canty & Ripley (2021), `https://cran.r-project.org/package=boot`

Examples

```
## Not run:
robust_regression(
  data = mtcars, formula = mpg ~ cyl * hp,
  iterations = 100
)

## End(Not run)
```

`round_flexibly` *Round flexibly*

Description

Round numbers to a flexible number of significant digits. "Flexible" rounding refers to rounding all numbers to the highest level of precision seen among numbers that would have resulted from the 'signif()' function in base R. The usage examples of this function demonstrate flexible rounding (see below).

Usage

```
round_flexibly(x = NULL, sigfigs = 3)
```

Arguments

`x`	a numeric vector
`sigfigs`	number of significant digits to flexibly round to. By default, `sigfigs = 3`.

Value

the output will be a numeric vector with values rounded to the highest level of precision seen among numbers that result from the 'signif()' function in base R.

Examples

```
# Example 1
# First, observe results from the 'signif' function:
c(0.00012345, pi)
signif(c(0.00012345, pi), 3)
# In the result above, notice how info is lost on some digits
# (e.g., 3.14159265 becomes 3.140000).
# In contrast, flexible rounding retains the lost info in the digits
round_flexibly(x = c(0.00012345, pi), sigfigs = 3)

# Example 2
# Again, first observe results from the 'signif' function:
c(0.12345, 1234, 0.12, 1.23, .01)
signif(c(0.12345, 1234, 0.12, 1.23, .01), 3)
# In the result above, notice how info is lost on some digits
# (e.g., 1234 becomes 1230.000).
# In contrast, flexible rounding retains the lost info in the digits.
# Specifically, in the example below, 0.12345 rounded to 3 significant
# digits (default) is signif(0.12345, 3) = 0.123 (3 decimal places).
# Because this 3 decimal places is the highest precision seen among
# all numbers, all other numbers will also be rounded to 3 decimal places.
round_flexibly(
c(0.12345, 1234, 0.12, 1.23, .01))

# Example 3
# If the input is a character vector, the original input will be returned.
round_flexibly(c("a", "b", "c"))

# Example 4
# If the input is a list (e.g., a data.frame) that contains at least
# one numeric vector, the numeric vector element(s) will be rounded
# flexibly.
round_flexibly(data.frame(a = c(1.2345, 123.45), b = c("a", "b")))

# Example 5
# If the input is a matrix, all numbers will be rounded flexibly
round_flexibly(matrix(
c(1.23, 2.345, 3.4567, 4.56789), ncol = 2), sigfigs = 3)
```

`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,
  dot_label_var_name = NULL,
  weight_var_name = NULL,
  alpha = 1,
  annotate_stats = TRUE,
  annotate_y_pos_rel = 5,
  annotate_y_pos_abs = NULL,
  annotated_stats_color = "green4",
  annotated_stats_font_size = 6,
  annotated_stats_font_face = "bold",
  line_of_fit_type = "lm",
  ci_for_line_of_fit = FALSE,
  line_of_fit_color = "blue",
  line_of_fit_thickness = 1,
  dot_color = "black",
  x_axis_label = NULL,
  y_axis_label = NULL,
  x_axis_tick_marks = NULL,
  y_axis_tick_marks = NULL,
  dot_size = 2,
  dot_label_size = NULL,
  dot_size_range = c(3, 12),
  jitter_x_y_percent = 0,
  jitter_x_percent = 0,
  jitter_y_percent = 0,
  cap_axis_lines = TRUE,
  color_dots_by = NULL,
  png_name = NULL,
  save_as_png = FALSE,
  width = 13,
  height = 9
)
```

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
`dot_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 (default = TRUE)
`annotate_y_pos_rel`	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). This value will be determined relative to the data. If `annotate_y_pos_rel = 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.
`annotate_y_pos_abs`	as an alternative to the argument `annotate_y_pos_rel`, the input for this argument will determine the position of the annotated stats. If `annotate_y_pos_abs = 7.5`, then the annotated stats will be placed at the y coordinate of 7.5. By default, this argument will be ignored unless it receives an input. That is, by default, the function will use the default value of the `annotate_y_pos_rel` argument to determine the y coordinate of the annotated stats.
`annotated_stats_color`	color of the annotated stats (default = "green4").
`annotated_stats_font_size`	font size of the annotated stats (default = 6).
`annotated_stats_font_face`	font face of the annotated stats (default = "bold").
`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
`line_of_fit_color`	color of the line of fit (default = "blue")
`line_of_fit_thickness`	thickness of the line of fit (default = 1)
`dot_color`	color of the dots (default = "black")

`x_axis_label`	alternative label for the x axis
`y_axis_label`	alternative label for the y axis
`x_axis_tick_marks`	a numeric vector indicating the positions of the tick marks on the x axis
`y_axis_tick_marks`	a numeric vector indicating the positions of the tick marks on the y axis
`dot_size`	size of the dots on the plot (default = 2)
`dot_label_size`	size for dots' labels on the plot. If no input is entered for this argument, it will be set as `dot_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 `dot_size_range`
`dot_size_range`	minimum and maximum size for dots on the plot when they are weighted
`jitter_x_y_percent`	horizontally and vertically jitter dots by a percentage of the respective ranges of x and y values.
`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)
`color_dots_by`	name of the variable that will determine colors of the dots
`png_name`	name of the PNG file to be saved. By default, the name will be "scatterplot_" followed by a timestamp of the current time. The timestamp will be in the format, jan_01_2021_1300_10_000001, where "jan_01_2021" would indicate January 01, 2021; 1300 would indicate 13:00 (i.e., 1 PM); and 10_000001 would indicate 10.000001 seconds after the hour.
`save_as_png`	if `save = TRUE`, the plot will be saved as a PNG file.
`width`	width of the plot to be saved. This argument will be directly entered as the `width` argument for the `ggsave` function within `ggplot2` package (default = 16)
`height`	height of the plot to be saved. This argument will be directly entered as the `height` argument for the `ggsave` function within `ggplot2` package (default = 9)

Details

If a weighted correlation is to be calculated, the following package(s) must be installed prior to running the function: Package 'weights' v1.0 (or possibly a higher version) by John Pasek (2018), `https://cran.r-project.org/package=weights`

Value

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

Examples

```
## Not run:
scatterplot(data = mtcars, x_var_name = "wt", y_var_name = "mpg")
scatterplot(
  data = mtcars, x_var_name = "wt", y_var_name = "mpg",
  dot_label_var_name = "hp", weight_var_name = "drat",
  annotate_stats = TRUE)
scatterplot(
  data = mtcars, x_var_name = "wt", y_var_name = "mpg",
  dot_label_var_name = "hp", weight_var_name = "cyl",
  dot_label_size = 7, annotate_stats = TRUE)
scatterplot(
data = mtcars, x_var_name = "wt", y_var_name = "mpg",
color_dots_by = "gear")

## End(Not run)
```

---

score_scale_items *Score scale items*

Description

Score items in a scale (e.g., Likert scale items) by computing the sum or mean of the items.

Usage

```
score_scale_items(
  item_list = NULL,
  reverse_item_list = NULL,
  operation = "mean",
  na.rm = FALSE,
  na_summary = TRUE,
  reverse_code_minuend = NULL
)
```

Arguments

`item_list`	a list of scale items (i.e., list of vectors of ratings) to code normally (as opposed to reverse coding).
`reverse_item_list`	a list of scale items to reverse code.
`operation`	if `operation = "mean"`, mean of the scale items will be calculated; if `operation = "sum"`, sum of the scale items will be calculated (default = "mean").
`na.rm`	logical. The `na.rm` argument that will be passed onto the base R's `rowMeans` or `rowSums` function (default = FALSE).
`na_summary`	logical. If `na_summary = TRUE` a summary of NA values will be printed; if `na_summary = FALSE` the summary will not be printed (default = TRUE).

`reverse_code_minuend`
: required for reverse coding; the number from which to subtract item ratings when reverse-coding. For example, if the items to reverse code are measured on a 7-point scale, enter `reverse_code_minuend = 8`.

Examples

```
score_scale_items(item_list = list(1:5, rep(3, 5)),
reverse_item_list = list(rep(5, 5)), reverse_code_minuend = 6)
score_scale_items(item_list = list(c(1, 1), c(1, 5)),
reverse_item_list = list(c(5, 3)),
reverse_code_minuend = 6, na_summary = FALSE)
score_scale_items(item_list = list(c(1, 1), c(1, 5)),
reverse_item_list = list(c(5, 1)),
reverse_code_minuend = 6, operation = "sum")
score_scale_items(item_list = list(1:5, rep(3, 5)))
score_scale_items(item_list = list(c(1, NA, 3), c(NA, 2, 3)))
score_scale_items(item_list = list(c(1, NA, 3), c(NA, 2, 3)), na.rm = TRUE)
```

`setup_r_env` *Set up R environment*

Description

Set up R environment by (1) clearing the console; (2) removing all objects in the global environment; (3) setting the working directory to the active document (in RStudio only); (4) unloading and loading the kim package.

Usage

```
setup_r_env(
  clear_console = TRUE,
  clear_global_env = TRUE,
  setwd_to_active_doc = TRUE,
  prep_kim = TRUE
)
```

Arguments

`clear_console`	if `TRUE`, clear the console (default = TRUE)
`clear_global_env`	if `TRUE`, remove all objects in the global environment (default = TRUE)
`setwd_to_active_doc`	if `TRUE`, set the working directory to the active document in RStudio (default = TRUE)
`prep_kim`	if `TRUE`, unload and load the kim package (default = TRUE)

Examples

```
## Not run:
setup_r_env()

## End(Not run)
```

---

setwd_to_active_doc *Set working directory to active document in RStudio*

Description

Set working directory to location of the active document in RStudio

Usage

```
setwd_to_active_doc()
```

Value

there will be no output from this function. Rather, the working directory will be set as location of the active document.

Examples

```
## Not run:
setwd_to_active_doc()

## End(Not run)
```

---

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`	Deprecated. By default, 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))
```

se_of_percentage *Standard Error (SE) of a percentage*

Description

Calculate the standard error of a percentage. See Fowler, Jr. (2014, p. 34, ISBN: 978-1-4833-1240-8)

Usage

```
se_of_percentage(percent = NULL, n = NULL)
```

Arguments

`percent`	a vector of percentages; each of the percentage values must be between 0 and 100
`n`	a vector of sample sizes; number of observations used to calculate each of the percentage values

Examples

```
se_of_percentage(percent = 40, n = 50)
se_of_percentage(percent = 50, n = 10)
```

se_of_proportion *Standard Error (SE) of a proportion*

Description

Calculate the standard error of a proportion. See Anderson and Finn (1996, p. 364, ISBN: 978-1-4612-8466-6)

Usage

```
se_of_proportion(p = NULL, n = NULL)
```

Arguments

`p`	a vector of proportions; each of the proportion values must be between 0 and 1
`n`	a vector of sample sizes; number of observations used to calculate each of the percentage values

Examples

```
se_of_proportion(p = 0.56, n = 400)
se_of_proportion(p = 0.5, n = 10)
```

`simple_effects_analysis` *Simple Effects Analysis*

Description

Conduct a simple effects analysis to probe a two-way interaction effect. See Field et al. (2012, ISBN: 978-1-4462-0045-2).

Usage

```
simple_effects_analysis(
  data = NULL,
  dv_name = NULL,
  iv_1_name = NULL,
  iv_2_name = NULL,
  iv_1_levels = NULL,
  iv_2_levels = NULL,
  print_contrast_table = "weights_sums_and_products",
  output = NULL
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`dv_name`	name of the dependent variable (DV)
`iv_1_name`	name of the first independent variable (IV1), whose main effects will be examined in the first set of contrasts
`iv_2_name`	name of the second independent variable (IV2), whose simple effects at each level of IV1 will be examined in the second set of contrasts
`iv_1_levels`	ordered levels of IV1
`iv_2_levels`	ordered levels of IV2
`print_contrast_table`	If `print_contrast_table = "weights_sums_and_products"`, contrasts' weights, sums of the weights and products will be printed. If `print_contrast_table = "weights_only"`, only the contrasts will be printed.

`output`	output can be one of the following: `"lm_object"`, `"table"`, `"weights_only"`, `"weights_sums_and_products"`, `"all"` By default, `output` = `NULL`, and there will be no output from the function other than the tables of simple effects and constrasts which will be printed on the console by default.

Value

By default, the function will print a table of contrasts and a table of simple effects.

Examples

```
factorial_anova_2_way(
  data = mtcars, dv_name = "mpg", iv_1_name = "vs",
  iv_2_name = "am", iterations = 100, plot = TRUE)
simple_effects_analysis(
  data = mtcars, dv_name = "mpg", iv_1_name = "vs",
  iv_2_name = "am")
```

`simple_slopes_analysis` *Simple slopes analysis*

Description

Conduct a simple slopes analysis, typically to probe a two-way interaction.

Usage

```
simple_slopes_analysis(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  mod_name = NULL,
  round_b = 2,
  round_se = 2,
  round_t = 2,
  round_p = 3,
  focal_values = NULL
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name`	name of the independent variable (IV)
`dv_name`	name of the dependent variable (DV)
`mod_name`	name of the moderator variable (MOD)

`round_b`	number of decimal places to which to round coefficients from the regression analysis (default = 2)
`round_se`	number of decimal places to which to round standard error values from the regression analysis (default = 2)
`round_t`	number of decimal places to which to round t statistics from the regression analysis (default = 2)
`round_p`	number of decimal places to which to round p values from the regression analysis (default = 2)
`focal_values`	this input will be used only in cases where IV is continuous. In such cases, what are the focal values of the IV at which to estimate the effect of MOD on DV? By default, values corresponding to the mean of IV +/-1 SD will be used.

Examples

```
simple_slopes_analysis(
data = mtcars, iv_name = "vs", dv_name = "mpg", mod_name = "hp")
simple_slopes_analysis(
data = mtcars, iv_name = "disp", dv_name = "mpg", mod_name = "hp")
simple_slopes_analysis(
data = mtcars, iv_name = "vs", dv_name = "am", mod_name = "hp")
simple_slopes_analysis(
data = mtcars, iv_name = "disp", dv_name = "am", mod_name = "hp")
```

---

simple_slopes_analysis_logistic
Simple slopes analysis with logistic regression analyses

Description

Conduct a simple slopes analysis with logistic regression analyses, typically to probe a two-way interaction when the dependent variable is binary.

Usage

```
simple_slopes_analysis_logistic(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  mod_name = NULL,
  round_b = 2,
  round_se = 2,
  round_z = 2,
  round_p = 3,
  focal_values = NULL
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name`	name of the independent variable (IV)
`dv_name`	name of the dependent variable (DV)
`mod_name`	name of the moderator variable (MOD)
`round_b`	number of decimal places to which to round coefficients from the regression analysis (default = 2)
`round_se`	number of decimal places to which to round standard error values from the regression analysis (default = 2)
`round_z`	number of decimal places to which to round t statistics from the regression analysis (default = 2)
`round_p`	number of decimal places to which to round p values from the regression analysis (default = 2)
`focal_values`	this input will be used only in cases where IV is continuous. In such cases, what are the focal values of the IV at which to estimate the effect of MOD on DV? By default, values corresponding to the mean of IV +/-1 SD will be used.

Examples

```
simple_slopes_analysis_logistic(
data = mtcars, iv_name = "vs", dv_name = "am", mod_name = "hp")
simple_slopes_analysis_logistic(
data = mtcars, iv_name = "disp", dv_name = "am", mod_name = "hp")
```

`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
kim::skewness(c(1, 2, 3, 4, 5, 10))
# calculate the traditional Fisher-Pearson coefficient of skewness
kim::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
kim::skewness(c(1, 2, 3, 4, 5, 10), type = "pearson_2")
```

---

spotlight_2_by_continuous
Spotlight 2 by Continuous

Description

Conduct a spotlight analysis for a 2 x Continuous design. See Spiller et al. (2013) doi:10.1509/jmr.12.0420

Usage

```
spotlight_2_by_continuous(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  mod_name = NULL,
  logistic = NULL,
  covariate_name = NULL,
  focal_values = NULL,
  interaction_p_include = TRUE,
  iv_level_order = NULL,
  output_type = "plot",
  colors = c("red", "blue"),
  dot_size = 3,
  observed_dots = FALSE,
  reg_lines = FALSE,
  reg_line_width = 1,
  reg_line_size = 1,
  lines_connecting_est_dv = TRUE,
  lines_connecting_est_dv_width = 1,
  estimated_dv_dot_shape = 15,
  estimated_dv_dot_size = 6,
  error_bar = "ci",
  error_bar_range = 0.95,
  error_bar_tip_width = NULL,
  error_bar_tip_width_percent = 8,
  error_bar_thickness = 1,
  error_bar_offset = NULL,
  error_bar_offset_percent = 8,
  simp_eff_bracket_leg_ht = NULL,
  simp_eff_bracket_leg_ht_perc = 2,
  simp_eff_bracket_offset = NULL,
  simp_eff_bracket_offset_perc = 1,
  simp_eff_bracket_color = "black",
  simp_eff_bracket_line_width = 1,
  simp_eff_text_offset = NULL,
  simp_eff_text_offset_percent = 7,
  simp_eff_text_hjust = 0.5,
  simp_eff_text_part_1 = "Simple Effect\n",
  simp_eff_text_color = "black",
  simp_eff_font_size = 5,
  interaction_p_value_x = NULL,
  interaction_p_value_y = NULL,
  interaction_p_value_font_size = 6,
  interaction_p_value_vjust = -1,
  interaction_p_value_hjust = 0.5,
  x_axis_breaks = NULL,
  x_axis_limits = NULL,
  x_axis_tick_mark_labels = NULL,
  y_axis_breaks = NULL,
  y_axis_limits = NULL,
  x_axis_space_left_perc = 10,
  x_axis_space_right_perc = 30,
  y_axis_tick_mark_labels = NULL,
  x_axis_title = NULL,
  y_axis_title = NULL,
  legend_title = NULL,
  legend_position = "right",
  y_axis_title_vjust = 0.85,
  round_decimals_int_p_value = 3,
  jitter_x_percent = 0,
  jitter_y_percent = 0,
  dot_alpha = 0.2,
  reg_line_alpha = 0.5,
  jn_point_font_size = 6,
  reg_line_types = c("solid", "dashed"),
  caption = NULL,
  plot_margin = ggplot2::unit(c(60, 30, 7, 7), "pt"),
  silent = FALSE
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`iv_name`	name of the binary independent variable (IV)
`dv_name`	name of the dependent variable (DV)
`mod_name`	name of the continuous moderator variable (MOD)
`logistic`	logical. Should logistic regressions be conducted, rather than ordinary least squares regressions? By default, ordinary least squares regressions will be conducted.
`covariate_name`	name(s) of the variable(s) to control for in estimating conditional values of the DV.
`focal_values`	focal values of the moderator variable at which to estimate IV's effect on DV.
`interaction_p_include`	logical. Should the plot include a p-value for the interaction term?
`iv_level_order`	order of levels in the independent variable for legend. By default, it will be set as levels of the independent variable ordered using R's base function `sort`.
`output_type`	type of output (default = "plot"). Other possible values include "spotlight_results", "dt_for_plotting", "modified_dt"
`colors`	set colors for the two levels of the independent variable By default, `colors = c("red", "blue")`.
`dot_size`	size of the observed_dots (default = 3)
`observed_dots`	logical. If `observed_dots = TRUE`, the observed values of all IV, DV, and MOD combinations will be plotted as dots. On top of these dots the spotlight analysis plot will be laid. If `observed_dots = FALSE`, these dots will not be plotted. By default, `observed_dots = FALSE`.
`reg_lines`	logical. If `reg_lines = TRUE`, the regression lines from regressing DV on MOD at each value of IV will be plotted. If `reg_lines = FALSE`, these regression lines will not be plotted. By default, `observed_dots = FALSE`.
`reg_line_width`	thickness of the regression lines (default = 1).
`reg_line_size`	deprecated. Use `reg_line_width` instead. thickness of the regression lines (default = 1).
`lines_connecting_est_dv`	logical. Should lines connecting the estimated values of DV be drawn? (default = TRUE)
`lines_connecting_est_dv_width`	thickness of the lines connecting the estimated values of DV (default = 1).
`estimated_dv_dot_shape`	ggplot value for shape of the dots at estimated values of DV (default = 15, a square shape).
`estimated_dv_dot_size`	size of the dots at estimated values of DV (default = 6).
`error_bar`	if `error_bar = "se"`; error bars will be +/-1 standard error, if `error_bar = "ci"` error bars will be a confidence interval. By default, `error_bar = "ci"`.

`error_bar_range`
: width of the confidence interval (default = 0.95 for a 95 percent confidence interval). This argument will not apply when `error_bar = "se"`

`error_bar_tip_width`
: graphically, width of the segments at the end of error bars (default = 0.13)

`error_bar_tip_width_percent`
: (default)

`error_bar_thickness`
: thickness of the error bars (default = 1)

`error_bar_offset`
: (default)

`error_bar_offset_percent`
: (default)

`simp_eff_bracket_leg_ht`
: (default)

`simp_eff_bracket_leg_ht_perc`
: (default)

`simp_eff_bracket_offset`
: (default)

`simp_eff_bracket_offset_perc`
: (default)

`simp_eff_bracket_color`
: (default)

`simp_eff_bracket_line_width`
: (default)

`simp_eff_text_offset`
: (default)

`simp_eff_text_offset_percent`
: (default)

`simp_eff_text_hjust`
: (default)

`simp_eff_text_part_1`
: The first part of the text for labeling simple effects. By default, `simp_eff_text_part_1 = "Simple Effect\n"`

`simp_eff_text_color`
: color for the text indicating p-values of simple effects (default = "black").

`simp_eff_font_size`
: font size of the text indicating p-values of simple effects (default = 5).

`interaction_p_value_x`
: (default)

`interaction_p_value_y`
: (default)

`interaction_p_value_font_size`
: font size for the interaction p value (default = 6)

`interaction_p_value_vjust`	(default)
`interaction_p_value_hjust`	(default)
`x_axis_breaks`	(default)
`x_axis_limits`	(default)
`x_axis_tick_mark_labels`	(default)
`y_axis_breaks`	(default)
`y_axis_limits`	(default)
`x_axis_space_left_perc`	(default)
`x_axis_space_right_perc`	(default)
`y_axis_tick_mark_labels`	(default)
`x_axis_title`	title of the x axis. By default, it will be set as input for `mod_name`. If `x_axis_title = FALSE`, it will be removed.
`y_axis_title`	title of the y axis. By default, it will be set as input for `dv_name`. If `y_axis_title = FALSE`, it will be removed.
`legend_title`	title of the legend. By default, it will be set as input for `iv_name`. If `legend_title = FALSE`, it will be removed.
`legend_position`	position of the legend (default = "right"). If `legend_position = "none"`, the legend will be removed.
`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.
`round_decimals_int_p_value`	To how many digits after the decimal point should the p value for the interaction term be rounded? (default = 3)
`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
`dot_alpha`	opacity of the dots (0 = completely transparent, 1 = completely opaque). By default, `dot_alpha = 0.2`
`reg_line_alpha`	(default)
`jn_point_font_size`	(default)
`reg_line_types`	types of the regression lines for the two levels of the independent variable. By default, `reg_line_types = c("solid", "dashed")`

`caption`	(default)
`plot_margin`	margin for the plot By default `plot_margin = ggplot2::unit(c(60, 30, 7, 7), "pt")`
`silent`	If `silent = FALSE`, (various) messages will be printed. If `silent = TRUE`, the messages will be suppressed. By default, `silent = FALSE`.

Examples

```
spotlight_2_by_continuous(
data = mtcars,
iv_name = "am",
dv_name = "mpg",
mod_name = "qsec")
# control for variables
spotlight_2_by_continuous(
data = mtcars,
iv_name = "am",
dv_name = "mpg",
mod_name = "qsec",
covariate_name = c("cyl", "hp"))
# control for variables and adjust simple effect labels
spotlight_2_by_continuous(
data = mtcars,
iv_name = "am",
dv_name = "mpg",
mod_name = "qsec",
covariate_name = c("cyl", "hp"),
reg_lines = TRUE,
observed_dots = TRUE,
error_bar_offset_percent = 3,
error_bar_tip_width_percent = 3,
simp_eff_text_offset_percent = 3,
simp_eff_bracket_leg_ht_perc = 2,
dot_alpha = 0.2,
simp_eff_text_part_1 = "")
# spotlight at specific values
spotlight_2_by_continuous(
data = mtcars,
iv_name = "am",
dv_name = "mpg",
mod_name = "qsec",
covariate_name = c("cyl", "hp"),
focal_values = seq(15, 22, 1),
reg_lines = TRUE,
observed_dots = TRUE,
dot_alpha = 0.2,
simp_eff_text_part_1 = "",
simp_eff_font_size = 4,
error_bar_offset_percent = 3,
error_bar_tip_width_percent = 3,
simp_eff_text_offset_percent = 3,
simp_eff_bracket_leg_ht_perc = 1,
x_axis_breaks = seq(15, 22, 1))
# spotlight for logistic regression
spotlight_2_by_continuous(
data = mtcars,
iv_name = "am",
dv_name = "vs",
mod_name = "drat",
logistic = TRUE)
```

standardize *Standardize*

Description

Standardize (i.e., normalize, obtain z-scores, or obtain the standard scores)

Usage

```
standardize(x = NULL)
```

Arguments

x	a numeric vector

Value

the output will be a vector of the standard scores of the input.

Examples

```
standardize(1:10)
```

standardized_regression *Standardized Regression*

Description

This function standardizes all variables for a regression analysis (i.e., dependent variable and all independent variables) and then conducts a regression with the standardized variables.

Usage

```
standardized_regression(
  data = NULL,
  formula = NULL,
  reverse_code_vars = NULL,
  sigfigs = NULL,
  round_digits_after_decimal = NULL,
  round_p = 3,
  pretty_round_p_value = TRUE,
  return_table_upper_half = FALSE,
  round_r_squared = 3,
  round_f_stat = 2,
  prettify_reg_table_col_names = TRUE
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`formula`	a formula object for the regression equation
`reverse_code_vars`	names of binary variables to reverse code
`sigfigs`	number of significant digits to round to
`round_digits_after_decimal`	round to nth digit after decimal (alternative to `sigfigs`)
`round_p`	number of decimal places to which to round p-values (default = 3)
`pretty_round_p_value`	logical. Should the p-values be rounded in a pretty format (i.e., lower threshold: "<.001"). By default, `pretty_round_p_value = TRUE`.
`return_table_upper_half`	logical. Should only the upper part of the table be returned? By default, `return_table_upper_half = FALSE`.
`round_r_squared`	number of digits after the decimal both r-squared and adjusted r-squared values should be rounded to (default 3)
`round_f_stat`	number of digits after the decimal the f statistic of the regression model should be rounded to (default 2)
`prettify_reg_table_col_names`	logical. Should the column names of the regression table be made pretty (e.g., change "std_beta" to "Std. Beta")? (Default = `TRUE`)

Value

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

Examples

```
standardized_regression(data = mtcars, formula = mpg ~ gear * cyl)
standardized_regression(
data = mtcars, formula = mpg ~ gear + gear:am + disp * cyl,
round_digits_after_decimal = 3)
```

`start_kim` *Start kim*

Description

Start kim (update kim; attach default packages; set working directory, etc.) 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

```
start_kim(
  update = TRUE,
  upgrade_other_pkg = FALSE,
  setup_r_env = TRUE,
  default_packages = c("data.table", "ggplot2"),
  silent_load_pkgs = c("data.table", "ggplot2")
)
```

Arguments

`update`	If `update = "force"`, force updating the package 'kim'. If `update = TRUE`, compares the currently installed package 'kim' with the most recent version on GitHub and, if the version on GitHub is more recent, ask the user to confirm the update. If confirmed, then update the package. If `update = FALSE`, skip updating the package. By default, `update = "force"`
`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`.
`setup_r_env`	logical. If `update = TRUE`, runs the function setup_r_env in the package "kim". Type "?kim::setup_r_env" to learn more. By default, `setup_r_env = TRUE`
`default_packages`	a vector of names of packages to load and attach. By default, `default_packages = c("data.table", "ggplot2")`

`silent_load_pkgs`	a character vector indicating names of packages to load silently (i.e., suppress messages that get printed when loading the packages). By default, `silent_load_pkgs = c("data.table", "ggplot2")`

Examples

```
## Not run:
start_kim()
start_kim(default_packages = c("dplyr", "ggplot2"))
start_kim(update = TRUE, setup_r_env = FALSE)

## End(Not run)
```

`su` *su: Sorted unique values*

Description

Extract unique elements and sort them

Usage

```
su(x = NULL, na.last = TRUE, decreasing = FALSE)
```

Arguments

`x`	a vector or a data frame or an array or NULL.
`na.last`	an argument to be passed onto the 'sort' function (in base R) for controlling the treatment of NA values. If `na.last = TRUE`, NA values in the data are put last; if `na.last = FALSE`, NA values are put first; if `na.last = NA`, NA values are removed. By default, `na.last = TRUE`
`decreasing`	logical. Should the sort be increasing or decreasing? An argument to be passed onto the 'sort' function (in base R). By default, `decreasing = FALSE`

Value

a vector, data frame, or array-like 'x' but with duplicate elements/rows removed.

Examples

```
su(c(10, 3, 7, 10, NA))
su(c("b", "z", "b", "a", NA, NA, NA))
```

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"
)
```

`tau_squared` *Tau-squared (between-studies variance for meta analysis)*

Description

Calculate tau-squared, the between-studies variance (the variance of the effect size parameters across the population of studies), as illustrated in Borenstein et al. (2009, pp. 72-73, ISBN: 978-0-470-05724-7).

Usage

```
tau_squared(effect_sizes = NULL, effect_size_variances = NULL)
```

Arguments

`effect_sizes`	effect sizes (e.g., standardized mean differences)
`effect_size_variances`	within-study variances

Details

Negative values of tau-squared are converted to 0 in the output (see Cheung, 2013; https://web.archive.org/web/2023051222255

Examples

```
## Not run:
tau_squared(effect_sizes = c(1, 2), effect_size_variances = c(3, 4))
# a negative tau squared value is converted to 0:
tau_squared(effect_sizes = c(1.1, 1.4), effect_size_variances = c(1, 4))

## End(Not run)
```

`theme_kim` *Theme Kim*

Description

A custom ggplot theme

Usage

```
theme_kim(
  legend_position = "none",
  legend_spacing_y = 1,
  legend_key_size = 3,
  base_size = 20,
  axis_tick_font_size = 20,
  axis_tick_marks_color = "black",
  axis_title_font_size = 24,
  y_axis_title_vjust = 0.85,
  axis_title_margin_size = 24,
  cap_axis_lines = FALSE
)
```

Arguments

`legend_position`	position of the legend (default = "none")
`legend_spacing_y`	vertical spacing of the legend keys in the unit of "cm" (default = 1)
`legend_key_size`	size of the legend keys in the unit of "lines" (default = 3)
`base_size`	base font size
`axis_tick_font_size`	font size for axis tick marks
`axis_tick_marks_color`	color of the 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 = FALSE)

Details

If a axis lines are to be capped at the ends, the following package(s) must be installed prior to running the function: Package 'lemon' v0.4.4 (or possibly a higher version) by Edwards et al. (2020), https://cran.r-project.org/package=lemon

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()
```

`top_median_or_bottom` *Top, median, or bottom*

Description

Indicates whether each value in a vector belongs to top, median, or bottom

Usage

```
top_median_or_bottom(vector)
```

Arguments

`vector`	a numeric vector

Value

a character vector indicating whether each element in a vector belongs to "top", "median", or "bottom"

Examples

```
top_median_or_bottom(c(1, 2, 3, NA))
top_median_or_bottom(c(1, 2, 2, NA))
top_median_or_bottom(c(1, 1, 2, NA))
```

---

tv *Tabulate vector*

Description

Shows frequency and proportion of unique values in a table format. This function is a copy of the earlier function, tabulate_vector, in Package 'kim'

Usage

```
tv(
  vector = NULL,
  na.rm = FALSE,
  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. By default, `FALSE`.
`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

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

`two_way_anova` *Two-Way Factorial ANOVA*

Description

This function is deprecated. Use the function 'factorial_anova_2_way' instead.

Usage

```
two_way_anova(
  data = NULL,
  dv_name = NULL,
  iv_1_name = NULL,
  iv_2_name = NULL,
  iv_1_values = NULL,
  iv_2_values = NULL,
  sigfigs = 3,
  robust = FALSE,
  iterations = 2000,
  plot = TRUE,
  error_bar = "ci",
  error_bar_range = 0.95,
  error_bar_tip_width = 0.13,
  error_bar_thickness = 1,
  error_bar_caption = TRUE,
  line_colors = NULL,
  line_types = NULL,
  line_thickness = 1,
  dot_size = 3,
  position_dodge = 0.13,
  x_axis_title = NULL,
  y_axis_title = NULL,
  y_axis_title_vjust = 0.85,
  legend_title = NULL,
  legend_position = "right",
  output = "anova_table",
  png_name = NULL,
  width = 7000,
  height = 4000,
  units = "px",
  res = 300,
  layout_matrix = NULL
)
```

Arguments

`data`	a data object (a data frame or a data.table)
`dv_name`	name of the dependent variable
`iv_1_name`	name of the first independent variable
`iv_2_name`	name of the second independent variable
`iv_1_values`	restrict all analyses to observations having these values for the first independent variable
`iv_2_values`	restrict all analyses to observations having these values for the second independent variable
`sigfigs`	number of significant digits to which to round values in anova table (default = 3)
`robust`	if `TRUE`, conduct a robust ANOVA in addition.
`iterations`	number of bootstrap samples for robust ANOVA. The default is set at 2000, but consider increasing the number of samples to 5000, 10000, or an even larger number, if slower handling time is not an issue.
`plot`	if `TRUE`, print a plot and enable returning an output (default = TRUE)
`error_bar`	if `error_bar = "se"`; error bars will be +/-1 standard error; if `error_bar = "ci"` error bars will be a confidence interval
`error_bar_range`	width of the confidence interval (default = 0.95 for 95 percent confidence interval). This argument will not apply when `error_bar = "se"`

`error_bar_tip_width`	graphically, width of the segments at the end of error bars (default = 0.13)
`error_bar_thickness`	thickness of the error bars (default = 1)
`error_bar_caption`	should a caption be included to indicate the width of the error bars? (default = TRUE).
`line_colors`	colors of the lines connecting means (default = NULL) If the second IV has two levels, then by default, `line_colors = c("red", "blue")`
`line_types`	types of the lines connecting means (default = NULL) If the second IV has two levels, then by default, `line_types = c("solid", "dashed")`
`line_thickness`	thickness of the lines connecting group means, (default = 1)
`dot_size`	size of the dots indicating group means (default = 3)
`position_dodge`	by how much should the group means and error bars be horizontally offset from each other so as not to overlap? (default = 0.13)
`x_axis_title`	a character string for the x-axis title. If no input is entered, then, by default, the first value of `iv_name` will be used as the x-axis title.
`y_axis_title`	a character string for the y-axis title. If no input is entered, then, by default, `dv_name` will be used as the y-axis title.
`y_axis_title_vjust`	position of the y axis title (default = 0.85). By default, `y_axis_title_vjust = 0.85`, which means that the y axis title will be positioned at 85% of the way up from the bottom of the plot.
`legend_title`	a character for the legend title. If no input is entered, then, by default, the second value of `iv_name` will be used as the legend title. If `legend_title = FALSE`, then the legend title will be removed.
`legend_position`	position of the legend: `"none"`, `"top"`, `"right"`, `"bottom"`, `"left"`, `"none"` (default = `"right"`)
`output`	output type can be one of the following: `"anova_table"`, `"group_stats"`, `"plot"`, `"robust_anova_results"`, `"robust_anova_post_hoc_results"`, `"robust_anova_post_hoc` `"all"`
`png_name`	name of the PNG file to be saved. If `png_name = TRUE`, the name will be "two_way_anova_" followed by a timestamp of the current time. The timestamp will be in the format, jan_01_2021_1300_10_000001, where "jan_01_2021" would indicate January 01, 2021; 1300 would indicate 13:00 (i.e., 1 PM); and 10_000001 would indicate 10.000001 seconds after the hour.
`width`	width of the PNG file (default = 7000)
`height`	height of the PNG file (default = 4000)
`units`	the units for the `width` and `height` arguments. Can be `"px"` (pixels), `"in"` (inches), `"cm"`, or `"mm"`. By default, `units = "px"`.
`res`	The nominal resolution in ppi which will be recorded in the png file, if a positive integer. Used for units other than the default. If not specified, taken as 300 ppi to set the size of text and line widths.
`layout_matrix`	The layout argument for arranging plots and tables using the `grid.arrange` function.

Details

Conduct a two-way factorial analysis of variance (ANOVA).

The following package(s) must be installed prior to running this function: Package 'car' v3.0.9 (or possibly a higher version) by Fox et al. (2020), https://cran.r-project.org/package=car

If robust ANOVA is to be conducted, the following package(s) must be installed prior to running the function: Package 'WRS2' v1.1-1 (or possibly a higher version) by Mair & Wilcox (2021), https://cran.r-project.org/package=WRS2

Value

by default, the output will be "anova_table"

Examples

```
## Not run:
two_way_anova(
  data = mtcars, dv_name = "mpg", iv_1_name = "vs",
  iv_2_name = "am", iterations = 100)
anova_results <- two_way_anova(
  data = mtcars, dv_name = "mpg", iv_1_name = "vs",
  iv_2_name = "am", output = "all")
anova_results

## End(Not run)
```

---

t_test_pairwise *t-tests, pairwise*

Description

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

Usage

```
t_test_pairwise(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  sigfigs = 3,
  cohen_d = TRUE,
  cohen_d_w_ci = TRUE,
  adjust_p = "holm",
  bonferroni = NULL,
  mann_whitney = TRUE,
  t_test_stats = FALSE,
  t_test_df_decimals = 1,
  sd = FALSE,
  round_p = 3,
  anova = TRUE,
  round_f = 2
)
```

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
`cohen_d`	if `cohen_d = TRUE`, Cohen's d statistics will be included in the output data.table.
`cohen_d_w_ci`	if `cohen_d_w_ci = TRUE`, Cohen's d with 95% CI will be included in the output data.table.
`adjust_p`	the name of the method to use to adjust p-values. If `adjust_p = "holm"`, the Holm method will be used; if `adjust_p = "bonferroni"`, the Bonferroni method will be used. By default, `adjust_p = "holm"`
`bonferroni`	The use of this argument is deprecated. Use the 'adjust_p' argument instead. If `bonferroni = TRUE`, Bonferroni tests will be conducted for t-tests or Mann-Whitney tests.
`mann_whitney`	if `TRUE`, Mann-Whitney test results will be included in the output data.table. If `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 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.
`round_p`	number of decimal places to which to round p-values (default = 3)
`anova`	Should a one-way ANOVA be conducted and reported? (default = TRUE)
`round_f`	number of decimal places to which to round the f statistic (default = 2)

Value

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

Examples

```
## Not run:
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)

## End(Not run)
```

und *Undocumented functions*

Description

A collection of miscellaneous functions lacking documentations

Usage

```
und(fn, ...)
```

Arguments

`fn`	name of the function
`...`	arguments for the function

Value

the output will vary by function

Examples

```
# correlation
und(corr_text, x = 1:5, y = c(1, 2, 2, 2, 3))
# mean center
und(mean_center, 1:10)
# compare results with base function
scale(1:10, scale = TRUE)
# find the modes
und(mode, c(3, 3, 3, 1, 2, 2))
# return values that are not outliers
und(outlier_rm, c(12:18, 100))
kim::outlier(c(1:10, 100))
```

unload_user_installed_pkgs *Unload all user-installed packages*

Description

Unload all user-installed packages

Usage

```
unload_user_installed_pkgs(exceptions = NULL, force = FALSE, keep_kim = TRUE)
```

Arguments

`exceptions`	a character vector of names of packages to keep loaded
`force`	logical. Should a package be unloaded even though other attached packages depend on it? By default, `force` = `FALSE`
`keep_kim`	logical. If `keep_kim` = `FALSE`, Package 'kim' will be detached along with all other user-installed packages. If `keep_kim` = `TRUE`, Package 'kim' will not be detached. By default, `keep_kim` = `FALSE`

Examples

```
## Not run:
unload_user_installed_pkgs()

## End(Not run)
```

`update_kim` *Update the package 'kim'*

Description

Updates the current package 'kim' 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_kim(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 'kim' package by installing the most recent version of the package from GitHub.

Examples

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

## End(Not run)
```

---

var_of_log_odds_ratio_to_var_of_d
Convert variance of log odds ratio to variance of d

Description

Convert the variance of a log odds ratio to the variance of a Cohen'd (standardized mean difference), as illustrated in Borenstein et al. (2009, p. 47, ISBN: 978-0-470-05724-7)

Usage

```
var_of_log_odds_ratio_to_var_of_d(var_of_log_odds_ratio = NULL)
```

Arguments

var_of_log_odds_ratio	the variance of a log odds ratio (the input can be a vector of values)

Examples

```
## Not run:
var_of_log_odds_ratio_to_var_of_d(1)

## End(Not run)
```

---

var_of_percentage *Variance of a percentage*

Description

Calculate the variance of a percentage. See Fowler, Jr. (2014, p. 34, ISBN: 978-1-4833-1240-8)

Usage

```
var_of_percentage(percent = NULL, n = NULL)
```

Arguments

`percent`	a vector of percentages; each of the percentage values must be between 0 and 100
`n`	a vector of sample sizes; number of observations used to calculate each of the percentage values

Examples

```
var_of_percentage(percent = 40, n = 50)
var_of_percentage(percent = 50, n = 10)
```

`var_of_proportion` *Variance of a proportion*

Description

Calculate the variance of a proportion. See Anderson and Finn (1996, p. 364, ISBN: 978-1-4612-8466-6)

Usage

```
var_of_proportion(p = NULL, n = NULL)
```

Arguments

`p`	a vector of proportions; each of the proportion values must be between 0 and 1
`n`	a vector of sample sizes; number of observations used to calculate each of the percentage values

Examples

```
var_of_proportion(p = 0.56, n = 400)
var_of_proportion(p = 0.5, n = 100)
var_of_proportion(p = 0.4, n = 50)
var_of_proportion(p = c(0.5, 0.9), n = c(100, 200))
```

`vlookup` *Vlookup*

Description

Look up values in a reference data.table and return values associated with the looked-up values contained in the reference data.table

Usage

```
vlookup(
  lookup_values = NULL,
  reference_dt = NULL,
  col_name_for_lookup_values = NULL,
  col_name_for_output_values = NULL
)
```

Arguments

`lookup_values`	a vector of values to look up
`reference_dt`	a data.table containing the values to look up as well as values associated with the looked-up values that need to be returned.
`col_name_for_lookup_values`	in the reference data.table, name of the column containing `lookup_values`.
`col_name_for_output_values`	in the reference data.table, name of the column containing values to return (i.e., values associated with the looked-up values that will be the function's output)

Examples

```
vlookup(lookup_values = c(2.620, 2.875), reference_dt = mtcars[1:9, ],
col_name_for_lookup_values = "wt", col_name_for_output_values = "qsec")
```

`weighted_mean_effect_size` *Estimate the mean effect size in a meta analysis*

Description

Estimate the mean effect size in a meta analysis, as illustrated in Borenstein et al. (2009, pp. 73-74, ISBN: 978-0-470-05724-7)

Usage

```
weighted_mean_effect_size(
  effect_sizes = NULL,
  effect_size_variances = NULL,
  ci = 0.95,
  one_tailed = FALSE,
  random_vs_fixed = "random"
)
```

Arguments

`effect_sizes`	effect sizes (e.g., standardized mean differences)
`effect_size_variances`	within-study variances
`ci`	width of the confidence interval (default = 0.95)
`one_tailed`	logical. If `one_tailed = FALSE`, a two-tailed p-value will be calculated. If `one_tailed = TRUE`, a one-tailed p-value will be calculated (default = FALSE)
`random_vs_fixed`	If `random_vs_fixed = "random"`, the summary effect will be calculated under the random-effects model (default = "random").

Examples

```
## Not run:
weighted_mean_effect_size(
effect_sizes = c(1, 2), effect_size_variances = c(3, 4))
weighted_mean_effect_size(
effect_sizes = c(0.095, 0.277, 0.367, 0.664, 0.462, 0.185),
effect_size_variances = c(0.033, 0.031, 0.050, 0.011, 0.043, 0.023))
# if effect sizes have a variance of 0, they will be excluded from
# the analysis
weighted_mean_effect_size(
effect_sizes = c(1.1, 1.2, 1.3, 1.4),
effect_size_variances = c(1, 0, 0, 4))

## End(Not run)
```

`weighted_mean_r` *Weighted mean correlation*

Description

Calculate the weighted mean correlation coefficient for a given correlations and sample sizes. This function uses the Hedges-Olkin Method with random effects. See Field (2001) doi:10.1037/1082-989X.6.2.161

Usage

```
weighted_mean_r(r = NULL, n = NULL, ci = 0.95, sigfigs = 3, silent = FALSE)
```

Arguments

r	a (vector of) correlation coefficient(s)
n	a (vector of) sample size(s)
ci	width of the confidence interval. Input can be any value less than 1 and greater than or equal to 0. By default, `ci = 0.95`. If `ci = TRUE`, the default value of 0.95 will be used. If `ci = FALSE`, no confidence interval will be estimated.
sigfigs	number of significant digits to round to (default = 3)
silent	logical. If `silent = FALSE`, a message regarding the weighted mean correlation and its p-value and CI will be printed. If `silent = TRUE`, this message will be suppressed. By default, `silent = FALSE`.

Value

the output will be a list of vector of correlation coefficient(s).

Examples

```
weighted_mean_r(r = c(0.2, 0.4), n = c(100, 100))
weighted_mean_r(r = c(0.2, 0.4), n = c(100, 20000))
# example consistent with using MedCalc
weighted_mean_r(
r = c(0.51, 0.48, 0.3, 0.21, 0.6, 0.46, 0.22, 0.25),
n = c(131, 129, 155, 121, 111, 119, 112, 145))
```

`weighted_z` *Weighted z*

Description

Calculate the weighted z (for calculating weighted mean correlation). See p. 231 of the book Hedges & Olkin (1985), Statistical Methods for Meta-Analysis (ISBN: 0123363802).

Usage

```
weighted_z(z = NULL, n = NULL)
```

Arguments

z	a vector of z values
n	a vector of sample sizes which will be used to calculate the weights, which in turn will be used to calculate the weighted z.

Value

the output will be a weighted z value.

Examples

```
weighted_z(1:3, c(100, 200, 300))
weighted_z(z = c(1:3, NA), n = c(100, 200, 300, NA))
```

---

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")
```

write_csv *Write to a csv file*

Description

Write to a csv file

Usage

```
write_csv(data = NULL, name = NULL, timestamp = NULL)
```

Arguments

`data`	a data object (a data frame or a data.table)
`name`	a character string of the csv file name without the ".csv" extension. For example, if the csv file to write to is "myfile.csv", enter `name = "myfile"`
`timestamp`	logical. Should the timestamp be appended to the file name?

Value

the output will be a .csv file in the working directory, that is, an output from the data.table function, `fwrite`

Examples

```
## Not run:
write_csv(mtcars, "mtcars_from_write_csv")
write_csv(mtcars)

## End(Not run)
```

z_score *z score*

Description

Calculate z-scores (i.e., standardize or obtain the standard scores)

Usage

```
z_score(x = NULL, na.rm = TRUE)
```

Arguments

`x`	a numeric vector
`na.rm`	logical. If `na.rm = TRUE`, NA values in the vector will be removed before calculating z-scores (default = TRUE).

Value

the output will be a vector of z-scores.

Examples

```
z_score(1:10)
```

`z_to_r_transform` *Z to r transformation (Inverse of Fisher's Z transformation)*

Description

Perform the Z-to-r transformation (i.e., the inverse of Fisher's r-to-Z transformation) for given Z value(s).

Usage

```
z_to_r_transform(z = NULL)
```

Arguments

`z`	a (vector of) Z values

Value

the output will be a vector of correlation coefficient(s) that are the result(s) of the Z-to-r transformation.

Examples

```
z_to_r_transform(2.646652)
z_to_r_transform(z = -3:3)
```

Index