

# Package ‘kim’

February 19, 2021

**Title** Functions for Behavioral Science Researchers

**Version** 0.2.96

**Description** Miscellaneous functions to simplify and expedite analyses of experimental data. Examples include a function that plots sample means of groups in a factorial experimental design, a function that conducts robust regressions with bootstrapped samples, and a function that conducts robust two-way analysis of variance.

**License** GPL-3

**URL** <https://github.com/jinkim3/kim>

**BugReports** <https://github.com/jinkim3/kim/issues>

**Imports** boot, car, data.table, devtools, effsize, ggplot2, ggridges, interactions, lemon, lm.beta, mediation, moments, paran, rstudioapi, stats, utils, weights, WRS2

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**NeedsCompilation** no

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

**Maintainer** Jin Kim <[jin.m.kim@yale.edu](mailto:jin.m.kim@yale.edu)>

**Repository** CRAN

**Date/Publication** 2021-02-19 07:30:07 UTC

## R topics documented:

barplot_for_counts . . . . .	3
capitalize . . . . .	3
chi_square_test_pairwise . . . . .	4
clean_data_from_qualtrics . . . . .	5
coeffcent_of_variation . . . . .	6
cohen_d . . . . .	7

cohen_d_from_cohen_textbook . . . . .	7
comma_sep_string_to_numbers . . . . .	8
compare_datasets . . . . .	9
compare_groups . . . . .	10
correlation_matrix . . . . .	10
cum_percent_plot . . . . .	11
desc_stats . . . . .	12
desc_stats_by_group . . . . .	13
find_duplicates . . . . .	14
floodlight_2_by_continuous . . . . .	15
ggsave_quick . . . . .	17
histogram . . . . .	18
histogram_by_group . . . . .	19
histogram_w_outlier_bins . . . . .	20
mann_whitney . . . . .	22
matrix_prep_dt . . . . .	23
mediation_analysis . . . . .	23
merge_data_tables . . . . .	24
merge_data_table_list . . . . .	25
multiple_regression . . . . .	26
order_rows_specifically_in_dt . . . . .	27
parallel_analysis . . . . .	28
percentile_rank . . . . .	28
plot_group_means . . . . .	29
population_variance . . . . .	30
prep . . . . .	31
pretty_round_p_value . . . . .	32
print_loop_progress . . . . .	33
proportion_of_values_in_vector . . . . .	34
read_csv . . . . .	34
read_sole_csv . . . . .	35
regex_match . . . . .	36
rel_pos_of_value_in_vector . . . . .	36
rel_value_of_pos_in_vector . . . . .	37
robust_regression . . . . .	37
scatterplot . . . . .	38
setup_r_env . . . . .	40
se_of_mean . . . . .	41
start_kim . . . . .	41
tabulate_vector . . . . .	42
theme_kim . . . . .	44
top_median_or_bottom . . . . .	45
two_way_anova . . . . .	45
t_test_pairwise . . . . .	47
update_kim . . . . .	48
wilcoxon_rank_sum_test . . . . .	48
write_csv . . . . .	49

---

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(  
  data = data.frame(  
    cyl = names(table(mtcars$cyl)),  
    count = as.vector(table(mtcars$cyl))  
  ),  
  x = "cyl", y = "count"  
)
```

---

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 substring (default = 1)
end	ending position of the substring (default = 1)

**Value**

a character string or a character vector

**Examples**

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

---

```
chi_square_test_pairwise
```

*Chi square test, pairwise*

---

**Description**

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

**Usage**

```
chi_square_test_pairwise(
  data = NULL,
  iv_name = NULL,
  dv_name = NULL,
  focal_dv_value = NULL,
  percentages_only = NULL,
  counts_only = NULL,
  sigfigs = 3
)
```

**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)
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

**Examples**

```
chi_square_test_pairwise(data = mtcars, iv_name = "vs", dv_name = "am")
chi_square_test_pairwise(data = mtcars, iv_name = "vs", dv_name = "am",
percentages_only = TRUE)
```

---

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_respose_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\_respose\_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")
```

---

coefficient\_of\_variation

*Coefficient of variation*

---

### Description

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

### Usage

```
coefficient_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

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

---

cohen_d	<i>Cohen's d with confidence interval</i>
---------	---

---

### Description

Calculates Cohen's d using the 'effsize' package v0.8.1 by Torchiano (2020) <https://github.com/mtorchiano/effsize/>

### Usage

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

---

cohen_d_from_cohen_textbook	<i>Cohen's d from Jacob Cohen's textbook (1988)</i>
-----------------------------	---

---

### Description

Calculates Cohen's d as described in Jacob Cohen's textbook (1988), Statistical Power Analysis for the Behavioral Sciences, 2nd Edition

**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

**Examples**

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

---

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	<i>Compare data sets</i>
------------------	--------------------------

---

### 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_groups	<i>Compare groups</i>
----------------	-----------------------

---

### 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)
```

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

### Examples

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

---

correlation_matrix	<i>correlation matrix</i>
--------------------	---------------------------

---

### 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 = "r"
)
```

**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

**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"))
```

---

cum_percent_plot	<i>Cumulative percentage plot</i>
------------------	-----------------------------------

---

**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.

**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,
  ci = TRUE,
  pi = TRUE,
  notify_na_count = NULL,
  print_dt = TRUE
)
```

**Arguments**

vector            a numeric vector

output\_type      if output\_type = "vector", return a vector of descriptive statistics; if output\_type = "dt", return a data.table of descriptive statistics

sigfigs           number of significant digits to round to (default = 3)

ci                logical. Should 95% CI be included in the descriptive stats? (default = TRUE)

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

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

print\_dt         if TRUE, print the descriptive stats data.table

**Value**

a named numeric vector or a data.table

**Examples**

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

---

desc\_stats\_by\_group    *Descriptive statistics by group*

---

**Description**

Returns descriptive statistics by group

**Usage**

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

**Arguments**

`data`            a data object (a data frame or a data.table)  
`var_for_stats`   name of the variable for which descriptive statistics will be calculated  
`grouping_vars`   name(s) of grouping variables  
`sigfigs`        number of significant digits to round to  
`cols_to_round`   names of columns whose values will be rounded

**Value**

a data.frame

**Examples**

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

---

find_duplicates	<i>Find duplicated values in a vector</i>
-----------------	---

---

### 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.
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")
```

---

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,  
  interaction_p_include = TRUE,  
  iv_level_order = NULL,  
  output = "reg_lines_plot",  
  jitter_x_percent = 0,  
  jitter_y_percent = 0,  
  dot_alpha = 0.5,  
  dot_size = 4,  
  legend_position = "right",  
  reg_line_types = c("solid", "dashed"),  
  jn_line_types = c("solid", "solid"),  
  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  
)
```

## 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
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").

<code>jitter_x_percent</code>	horizontally jitter dots by a percentage of the range of x values
<code>jitter_y_percent</code>	vertically jitter dots by a percentage of the range of y values#'
<code>dot_alpha</code>	opacity of the dots (0 = completely transparent, 1 = completely opaque). By default, <code>dot_alpha = 0.5</code>
<code>dot_size</code>	size of the dots (default = 4)
<code>legend_position</code>	position of the legend (default = "right"). If <code>legend_position = "none"</code> , the legend will be removed.
<code>reg_line_types</code>	types of the regression lines for the two levels of the independent variable. By default, <code>reg_line_types = c("solid", "dashed")</code>
<code>jn_line_types</code>	types of the lines for Johnson-Neyman points. By default, <code>jn_line_types = c("solid", "solid")</code>
<code>sig_region_color</code>	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.
<code>sig_region_alpha</code>	opacity for <code>sig_region_color</code> . (0 = completely transparent, 1 = completely opaque). By default, <code>sig_region_alpha = 0.08</code>
<code>nonsig_region_color</code>	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.
<code>nonsig_region_alpha</code>	opacity for <code>nonsig_region_color</code> . (0 = completely transparent, 1 = completely opaque). By default, <code>nonsig_region_alpha = 0.08</code>
<code>x_axis_title</code>	title of the x axis. By default, it will be set as input for <code>mod_name</code> . If <code>x_axis_title = FALSE</code> , it will be removed.
<code>y_axis_title</code>	title of the y axis. By default, it will be set as input for <code>dv_name</code> . If <code>y_axis_title = FALSE</code> , it will be removed.
<code>legend_title</code>	title of the legend. By default, it will be set as input for <code>iv_name</code> . If <code>legend_title = FALSE</code> , it will be removed.

### Examples

```
floodlight_2_by_continuous(
  data = mtcars,
  iv_name = "am",
  dv_name = "mpg",
  mod_name = "qsec")
```



---

ggsave_quick	<i>ggsave quick</i>
--------------	---------------------

---

## Description

quickly save the current plot with a timestamp

## Usage

```
ggsave_quick(  
  name = NULL,  
  timestamp = NULL,  
  file_name_extension = ".png",  
  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
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.
file_name_extension	file name extension (default = ".png")
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

### Usage

```

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

```

### Arguments

<code>vector</code>	a numeric vector
<code>number_of_bins</code>	number of bins for the histogram (default = 30)
<code>x_tick_marks</code>	a vector of values at which to place tick marks on the x axis (e.g., setting <code>x_tick_marks = seq(0, 10, 5)</code> will put tick marks at 0, 5, and 10.)
<code>y_tick_marks</code>	a vector of values at which to place tick marks on the y axis (e.g., setting <code>y_tick_marks = seq(0, 10, 5)</code> will put tick marks at 0, 5, and 10.)
<code>fill_color</code>	color for inside of the bins (default = "cyan4")
<code>border_color</code>	color for borders of the bins (default = "black")
<code>y_axis_title_vjust</code>	position of the y axis title (default = 0.85).
<code>x_axis_title</code>	title for x axis (default = "Value")
<code>y_axis_title</code>	title for y axis (default = "Count")
<code>notify_na_count</code>	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

a ggplot object

**Examples**

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

---

histogram\_by\_group      *Histogram by group*

---

**Description**

Creates histograms by group to compare distributions

**Usage**

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

**Arguments**

data	a data object (a data frame or a data.table)
iv_name	name of the independent variable
dv_name	name of the dependent variable
order_of_groups_top_to_bot	a character vector indicating the desired presentation order of levels in the independent variable (from the top to bottom). Omitting a group in this argument will remove the group in the set of histograms.
number_of_bins	number of bins for the histograms (default = 40)
space_between_histograms	space between histograms (minimum = 0, maximum = 1, default = 0.15)
draw_baseline	logical. Should the baseline and the trailing lines to either side of the histogram be drawn? (default = FALSE)

**Value**

a ggplot object

**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")
```

---

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))
```

---

mann\_whitney

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

---

**Description**

A nonparametric equivalent of the independent t-test

**Usage**

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

**Arguments**

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

**Value**

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

**Examples**

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

---

matrix_prep_dt	<i>Prepare a two-column data.table that will be used to fill values in a matrix</i>
----------------	---

---

**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")  
)
```

---

mediation_analysis	<i>Mediation analysis</i>
--------------------	---------------------------

---

**Description**

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

**Usage**

```
mediation_analysis(  
  data = NULL,  
  iv_name = NULL,  
  mediator_name = NULL,  
  dv_name = NULL,  
  covariates_names = NULL,  
  robust_se = TRUE,  
  iterations = 1000  
)
```

**Arguments**

<code>data</code>	a data object (a data frame or a data.table)
<code>iv_name</code>	name of the independent variable
<code>mediator_name</code>	name of the mediator variable
<code>dv_name</code>	name of the dependent variable
<code>covariates_names</code>	names of covariates to control for
<code>robust_se</code>	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.
<code>iterations</code>	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.

**Examples**

```
mediation_analysis(
  data = mtcars, iv_name = "cyl",
  mediator_name = "disp", dv_name = "mpg", iterations = 100
)
```

---

<code>merge_data_tables</code>	<i>Merge data tables</i>
--------------------------------	--------------------------

---

**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**

<code>dt1</code>	the first data.table which will remain intact
<code>dt2</code>	the second data.table which will be joined outside of (around) the first data.table. If there are any duplicated ID values and column names across the two data tables, the cell values in the first data.table will remain intact and the cell values in the second data.table will be discarded for the resulting merged data table.



id	name of the column that will contain the ID values in the two data tables. The name of the ID column must be identical in the two data tables.
silent	If <code>silent = TRUE</code> , no message will be printed regarding how many ID values and column names were duplicated. If <code>silent = FALSE</code> , messages will be printed regarding how many ID values and column names were duplicated. (default = <code>FALSE</code> )

**Value**

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

**Examples**

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

---

merge\_data\_table\_list *Merge a list of data tables*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

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

**Examples**

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

---

multiple\_regression     *Summarize multiple regression results in a data.table*

---

**Description**

Summarize multiple regression results in a data.table

**Usage**

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

**Arguments**

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

**Examples**

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

---

```
order_rows_specifically_in_dt
                                  Order rows specifically in a data table
```

---

**Description**

Order rows in a data.table in a specific order

**Usage**

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

**Arguments**

dt            a data.table object  
 col\_to\_order\_by  
                  a character value indicating the name of the column by which to order the data.table  
 specific\_order a vector indicating a specific order of the values in the column by which to order the data.table.

**Value**

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))
```

---

parallel\_analysis      *Parallel analysis*

---

### Description

Conducts a parallel analysis to determine how many factors to retain in a factor analysis. Uses the 'paran' package v1.5.2 by Dinno (2018) <https://cran.r-project.org/package=paran>

### Usage

```
parallel_analysis(
  data = NULL,
  names_of_vars = NULL,
  iterations = NULL,
  percentile_for_eigenvalue = 95
)
```

### 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).

### 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)
```

---

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,
  lines_connecting_means = TRUE,
  line_size = 1,
  dot_size = 3,
  error_bar_tip_width = 0.13,
  error_bar_thickness = 1,
  position_dodge = 0.13,
  legend_position = "right",
  y_axis_title_vjust = 0.85
)
```

**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"
lines_connecting_means	logical. Should lines connecting means within each group be drawn? (default = TRUE)
line_size	thickness of the lines connecting group means, (default = 1)
dot_size	size of the dots indicating group means (default = 3)
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)
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_position	position of the legend: "none", "top", "right", "bottom", "left", "none" (default = "right")
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.

**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
)
```

---

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 prepara-
              tion of package(s) is successful. If silent_if_successful = FALSE, a mes-
              sage 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
```

## Examples

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

---

pretty\_round\_p\_value *Pretty round p-value*

---

## Description

Pretty round p-value

## Usage

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

## Arguments

`p_value_vector` one number or a numeric vector  
`round_digits_after_decimal`  
round to nth digit after decimal  
`include_p_equals`  
if TRUE, output will be a string of mathematical expression including "p", e.g.,  
"p < .01"

## Examples

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



---

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,  
  percent = 1  
)
```

## 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.
percent	if percent = 1, progress level will be printed at every 1 percent progress (default = 1)

## 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(0.2)  
  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, vector, na.exclude = TRUE)
```

### Arguments

values	a set of values
vector	a numeric or character vector
na.exclude	if TRUE, NA values will be removed both from vector and values before calculation

### Examples

```
proportion_of_values_in_vector(  
  values = 2:3, vector = c(1:3, NA)  
)  
proportion_of_values_in_vector(  
  values = 2:3, vector = c(1:3, NA), na.exclude = FALSE  
)  
proportion_of_values_in_vector(  
  values = c(2:3, NA), vector = c(1:3, NA), na.exclude = FALSE  
)
```

---

read\_csv

*Read a csv file*

---

### Description

Read a csv file

### Usage

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

**Arguments**

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

**Value**

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

**Examples**

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

## End(Not run)
```

---

read_sole_csv	<i>Read the sole csv file in the working directory</i>
---------------	--

---

**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	<i>Regular expression matches</i>
-------------	-----------------------------------

---

**Description**

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

**Usage**

```
regex_match(regex = NULL, vector = NULL, mute_report = 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
mute_report	logical. Should the report on regular expression matches be printed?
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	<i>Find relative position of a value in a vector</i>
----------------------------	--

---

**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)
```

---

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.

**Examples**

```
robust_regression(
  data = mtcars, formula = mpg ~ cyl * hp,
  iterations = 100
)
```

---

scatterplot

*Scatterplot*


---

**Description**

Creates a scatter plot and calculates a correlation between two variables

**Usage**

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

**Arguments**

<code>data</code>	a data object (a data frame or a data.table)
<code>x_var_name</code>	name of the variable that will go on the x axis
<code>y_var_name</code>	name of the variable that will go on the y axis
<code>point_label_var_name</code>	name of the variable that will be used to label individual observations
<code>weight_var_name</code>	name of the variable by which to weight the individual observations for calculating correlation and plotting the line of fit
<code>alpha</code>	opacity of the dots (0 = completely transparent, 1 = completely opaque)
<code>annotate_stats</code>	if TRUE, the correlation and p-value will be annotated at the top of the plot
<code>annotate_y_pos</code>	position of the annotated stats, expressed as a percentage of the range of y values by which the annotated stats will be placed above the maximum value of y in the data set (default = 5). If <code>annotate_y_pos = 5</code> , 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, $y = 0.05 * (100 - 0) + 100 = 105$ .
<code>line_of_fit_type</code>	if <code>line_of_fit_type = "lm"</code> , a regression line will be fit; if <code>line_of_fit_type = "loess"</code> , a local regression line will be fit; if <code>line_of_fit_type = "none"</code> , no line will be fit
<code>ci_for_line_of_fit</code>	if <code>ci_for_line_of_fit = TRUE</code> , confidence interval for the line of fit will be shaded
<code>x_axis_label</code>	alternative label for the x axis
<code>y_axis_label</code>	alternative label for the y axis
<code>point_label_size</code>	size for dots' labels on the plot. If no input is entered for this argument, it will be set as <code>point_label_size = 5</code> 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 <code>point_size_range</code>
<code>point_size_range</code>	minimum and maximum size for dots on the plot when they are weighted
<code>jitter_x_percent</code>	horizontally jitter dots by a percentage of the range of x values
<code>jitter_y_percent</code>	vertically jitter dots by a percentage of the range of y values
<code>cap_axis_lines</code>	logical. Should the axis lines be capped at the outer tick marks? (default = TRUE)

**Value**

a ggplot object

## Examples

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

---

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 current file; (4) unloading and loading the kim package

## Usage

```
setup_r_env(
  clear_console = TRUE,
  clear_global_env = TRUE,
  set_wd_to_current_file = 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)  
`set_wd_to_current_file` if TRUE, set the working directory to the current file (default = TRUE)  
`prep_kim` if TRUE, unload and load the kim package (default = TRUE)

## Examples

```
## Not run:
setup_r_env()

## End(Not run)
```



---

se_of_mean	<i>Standard error of the mean</i>
------------	-----------------------------------

---

**Description**

Standard error of the mean

**Usage**

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

**Arguments**

vector	a numeric vector
na.rm	if TRUE, NA values will be removed before calculation
notify_na_count	if TRUE, notify how many observations were removed due to missing values. By default, NA count will be printed only if there are any NA values.

**Examples**

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

---

start_kim	<i>Start kim</i>
-----------	------------------

---

**Description**

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

**Usage**

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

**Arguments**

`update` logical. If `update = TRUE`, updates the current package "kim" by installing the most recent version (probably from Github) By default, `update = TRUE`

`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 packaged). By default, `silent_load_pkgs = NULL`

**Examples**

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

## End(Not run)
```

---

<code>tabulate_vector</code>	<i>Tabulate vector</i>
------------------------------	------------------------

---

**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**

a data.table or data.frame

**Examples**

```

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

```

---

`theme_kim`*Theme Kim*

---

### Description

A custom ggplot theme

### Usage

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

### Arguments

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

`base_size` base font size

`axis_tick_font_size` font size for axis tick marks

`axis_title_font_size` font size for axis title

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

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

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

### Examples

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

---

top_median_or_bottom	<i>Top, median, or bottom</i>
----------------------	-------------------------------

---

**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))
```

---

two_way_anova	<i>Two-way ANOVA</i>
---------------	----------------------

---

**Description**

Two-way ANOVA

**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 = FALSE,
```

```

    error_bar = "ci",
    error_bar_range = 0.95,
    line_size = 1,
    dot_size = 3,
    error_bar_tip_width = 0.13,
    position_dodge = 0.13,
    legend_position = "right",
    output = NULL
)

```

### Arguments

<code>data</code>	a data object (a data frame or a data.table)
<code>dv_name</code>	name of the dependent variable
<code>iv_1_name</code>	name of the first independent variable
<code>iv_2_name</code>	name of the second independent variable
<code>iv_1_values</code>	restrict all analyses to observations having these values for the first independent variable
<code>iv_2_values</code>	restrict all analyses to observations having these values for the second independent variable
<code>sigfigs</code>	number of significant digits to which to round values in anova table (default = 3)
<code>robust</code>	if TRUE, conduct a robust ANOVA in addition.
<code>iterations</code>	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.
<code>plot</code>	if TRUE, print a plot and enable returning an output.
<code>error_bar</code>	if <code>error_bar = "se"</code> ; error bars will be +/-1 standard error, if <code>error_bar = "ci"</code> error bars will be a confidence interval; if <code>error_bar = "pi"</code> , error bars will be a prediction interval
<code>error_bar_range</code>	width of the confidence or prediction interval (default = 0.95 for 95 percent confidence or prediction interval). This argument will not apply when <code>error_bar = "se"</code>
<code>line_size</code>	thickness of the lines connecting group means, (default = 1)
<code>dot_size</code>	size of the dots indicating group means (default = 3)
<code>error_bar_tip_width</code>	graphically, width of the segments at the end of error bars (default = 0.13)
<code>position_dodge</code>	by how much should the group means and error bars be horizontally offset from each other so as not to overlap? (default = 0.13)
<code>legend_position</code>	position of the legend: "none", "top", "right", "bottom", "left", "none" (default = "right")
<code>output</code>	output type can be one of the following: "anova_table", "group_stats", "plot", "levens_test_result", "robust_anova_results", "robust_anova_post_hoc_results", "robust_anova_post_hoc_contrast"

**Value**

by default, the output will be "anova\_table"

**Examples**

```
two_way_anova(
  data = mtcars, dv_name = "mpg", iv_1_name = "vs",
  iv_2_name = "am", iterations = 100
)
```

---

t_test_pairwise	<i>t test, pairwise</i>
-----------------	-------------------------

---

**Description**

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

**Usage**

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

**Arguments**

data	a data object (a data frame or a data.table)
iv_name	name of the independent variable
dv_name	name of the dependent variable
sigfigs	number of significant digits to round to
mann_whitney	if TRUE, Mann-Whitney test results will be included in the output data.table. If TRUE, Mann-Whitney tests will not be performed.

**Value**

the output will be a data.table

**Examples**

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

---

update_kim	<i>Update 'kim'</i>
------------	---------------------

---

**Description**

Updates the current package 'kim' by installing the most recent version

**Usage**

```
update_kim(source = "github", force = FALSE, upgrade = FALSE)
```

**Arguments**

source	location of the most recent version of the package (default = "github")
force	logical. If force = TRUE, forces the installation even if the package 'kim' from the source has not changed since last install.
upgrade	input for upgrade argument to be passed on to remotes::install_github. The default value is FALSE. An input could be TRUE or FALSE, but it could also be one of the following: "default", "ask", "always", or "never". "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 are also accepted and correspond to "always" and "never" respectively.

**Examples**

```
## Not run:
update_kim()

## End(Not run)
```

---

wilcoxon_rank_sum_test
------------------------

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

---

**Description**

A nonparametric equivalent of the independent t-test

**Usage**

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



**Arguments**

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

**Value**

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

**Examples**

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

---

write_csv	<i>Write to a csv file</i>
-----------	----------------------------

---

**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)
```

# Index

barplot\_for\_counts, 3

capitalize, 3

chi\_square\_test\_pairwise, 4

clean\_data\_from\_qualtrics, 5

coefficient\_of\_variation, 6

cohen\_d, 7

cohen\_d\_from\_cohen\_textbook, 7

comma\_sep\_string\_to\_numbers, 8

compare\_datasets, 9

compare\_groups, 10

correlation\_matrix, 10

cum\_percent\_plot, 11

desc\_stats, 12

desc\_stats\_by\_group, 13

find\_duplicates, 14

floodlight\_2\_by\_continuous, 15

ggsave\_quick, 17

histogram, 18

histogram\_by\_group, 19

histogram\_w\_outlier\_bins, 20

mann\_whitney, 22

matrix\_prep\_dt, 23

mediation\_analysis, 23

merge\_data\_table\_list, 25

merge\_data\_tables, 24

multiple\_regression, 26

order\_rows\_specifically\_in\_dt, 27

parallel\_analysis, 28

percentile\_rank, 28

plot\_group\_means, 29

population\_variance, 30

prep, 31

pretty\_round\_p\_value, 32

print\_loop\_progress, 33

proportion\_of\_values\_in\_vector, 34

read\_csv, 34

read\_sole\_csv, 35

regex\_match, 36

rel\_pos\_of\_value\_in\_vector, 36

rel\_value\_of\_pos\_in\_vector, 37

robust\_regression, 37

scatterplot, 38

se\_of\_mean, 41

setup\_r\_env, 40

start\_kim, 41

t\_test\_pairwise, 47

tabulate\_vector, 42

theme\_kim, 44

top\_median\_or\_bottom, 45

two\_way\_anova, 45

update\_kim, 48

wilcoxon\_rank\_sum\_test, 48

write\_csv, 49