Package ‘wrappedtools’

March 16, 2024

Type Package

Title Useful Wrappers Around Commonly Used Functions

Description The main functionalities of 'wrappedtools' are:
adding backticks to variable names; rounding to desired precision
with special case for p-values;
selecting columns based on pattern and storing their position, name,
and backticked name; computing and formatting of descriptive statistics
(e.g. mean±SD), comparing groups and creating publication-ready tables with
descriptive statistics and p-values; creating specialized plots for
correlation matrices. Functions were mainly written for my own daily work or
teaching, but may be of use to others as well.

Version 0.9.5

Date 2024-03-16

Maintainer Andreas Busjahn <andreas@busjahn.net>

License GPL-3

Encoding UTF-8

Imports stats, boot, knitr, coin, utils, dplyr, forcats, purrr, glue,
rlang, stringr, ggplot2, tibble, tidyr, kableExtra, lifecycle,
broom, rlist, DescTools, flextable

Depends R (>= 4.2)

RoxygenNote 7.3.1

LazyData true

VignetteBuilder knitr

Suggests rmarkdown, testthat, ggrepel

URL https://github.com/abusjahn/wrappedtools

BugReports https://github.com/abusjahn/wrappedtools/issues

NeedsCompilation no

Author Andreas Busjahn [cre, aut] (<https://orcid.org/0000-0001-9650-6919>),
Bilal Asser [aut]

Repository CRAN

Date/Publication 2024-03-16 17:40:02 UTC

R topics documented:

bt — *Add backticks to names or remove them*

Description

bt adds leading and trailing backticks to make illegal variable names usable. Optionally removes them.

Usage

```
bt(x, remove = FALSE)
```

Arguments

x	Names to add backtick to.
remove	Option to remove existing backticks, default=FALSE.

Value

Character vector with backticks added.

Examples

```
bt('name 1')
```

cat_desc_stats — *Compute absolute and relative frequencies.*

Description

cat_desc_stats computes absolute and relative frequencies for categorical data with a number of formatting options.

Usage

```
cat_desc_stats(
  source = NULL,
  separator = " ",
  return_level = TRUE,
  ndigit = 0,
  groupvar = NULL,
  singleline = FALSE,
  percent = TRUE,
  prettynum = FALSE,
  .german = FALSE,
  quelle = NULL
)
```

Arguments

`source`	Data for computation. Previously "quelle".
`separator`	delimiter between results per level, preset as ' '.
`return_level`	Should levels be reported?
`ndigit`	Digits for rounding of relative frequencies.
`groupvar`	Optional grouping factor.
`singleline`	Put all group levels in a single line?
`percent`	Logical, add percent-symbol after relative frequencies?
`prettynum`	logical, apply prettyNum to results?
`.german`	logical, should "." and "," be used as bigmark and decimal? Sets prettynum to TRUE.
`quelle`	deprecated, retained for compatibility, use 'source' instead.

Value

Structure depends on parameter return_level: if FALSE than a tibble with descriptives, otherwise a list with two tibbles with levels of factor and descriptives. If parameter singleline is FALSE (default), results for each factor level is reported in a separate line, otherwise they are pasted. Number of columns for result tibbles is one or number of levels of the additional grouping variable.

Examples

```
cat_desc_stats(mtcars$gear)
cat_desc_stats(mtcars$gear, return_level = FALSE)
cat_desc_stats(mtcars$gear, groupvar = mtcars$am)
cat_desc_stats(mtcars$gear, groupvar = mtcars$am, singleline = TRUE)
```

`cat_desc_table` *Compute absolute and relative frequencies for a table.*

Description

`cat_desc_table` computes absolute and relative frequencies for categorical data with a number of formatting options.

Usage

```
cat_desc_table(
  data,
  desc_vars,
  round_desc = 2,
  singleline = FALSE,
  spacer = " ",
  indentor = ""
)
```

Arguments

`data`	name of data set (tibble/data.frame) to analyze.
`desc_vars`	vector of column names for dependent variables.
`round_desc`	number of significant digits for rounding of descriptive stats.
`singleline`	Put all group levels in a single line?
`spacer`	Text element to indent levels and fill empty cells, defaults to " ".
`indentor`	Optional text to indent factor levels

Value

A tibble with variable names and descriptive statistics.

Examples

```
cat_desc_table(
  data = mtcars, desc_vars = c("gear", "cyl", "carb"))

cat_desc_table(
  data = mtcars, desc_vars = c("gear", "cyl", "carb"), singleline = TRUE)
```

`cn` *Shortcut for colnames()*

Description

`cn` lists column names, by default for variable rawdata.

Usage

```
cn(data = rawdata)
```

Arguments

`data`	Data structure to read column names from.

Value

Character vector with column names.

Examples

```
cn(mtcars)
```

`ColSeeker` *Find numeric index and names of columns based on type and patterns*

Description

`ColSeeker` looks up colnames (by default for tibble rawdata) based on type and parts of names, using regular expressions. Be warned that special characters as e.g. `[ (` need to be escaped or replaced by `.` Exclusion rules may be specified as well.

Usage

```
ColSeeker(
  data = rawdata,
  namepattern = ".",
  varclass = NULL,
  exclude = NULL,
  excludeclass = NULL,
  casesensitive = TRUE,
  returnclass = FALSE
)
```

Arguments

`data`	tibble or data.frame, where columns are to be found; by default rawdata
`namepattern`	Vector of pattern to look for.
`varclass`	Vector, only columns of defined class(es) are returned
`exclude`	Vector of pattern to exclude from found names.
`excludeclass`	Vector, exclude columns of specified class(es)
`casesensitive`	Logical if case is respected in matching (default FALSE: a<>A)
`returnclass`	Logical if classes should be included in output

Value

A list with index, names, and backticked names, optionally the classes as well

Examples

```
ColSeeker(data = mtcars, namepattern = c("^c", "g"))
ColSeeker(data = mtcars, namepattern = c("^c", "g"), exclude = "r")
```

compare2numvars *Comparison for columns of numbers for 2 groups*

Description

compare2numvars computes either t_var_test or wilcox.test, depending on parameter gaussian. Descriptive statistics, depending on distribution, are reported as well.

Usage

```
compare2numvars(
  data,
  dep_vars,
  indep_var,
  gaussian,
  round_p = 3,
  round_desc = 2,
  range = FALSE,
  rangesep = " ",
  pretext = FALSE,
  mark = FALSE,
  n = FALSE,
  add_n = FALSE
)
```

Arguments

data	name of dataset (tibble/data.frame) to analyze.
dep_vars	vector of column names for independent variables.
indep_var	name of grouping variable, has to translate to 2 groups. If more levels are encountered, an error is produced.
gaussian	logical specifying normal or ordinal values.
round_p	level for rounding p-value.
round_desc	number of significant digits for rounding of descriptive stats.
range	include min/max?
rangesep	text between statistics and range or other elements.
pretext	for function formatP.
mark	for function formatP.
n	create columns for n per group?
add_n	add n to descriptive statistics?

Value

A tibble with variable names, descriptive statistics, and p-value, number of rows is number of dep_vars.

Examples

```
# Assuming Normal distribution:
compare2numvars(
  data = mtcars, dep_vars = c("wt", "mpg", "qsec"), indep_var = "am",
  gaussian = TRUE
)
# Ordinal scale:
compare2numvars(
  data = mtcars, dep_vars = c("wt", "mpg", "qsec"), indep_var = "am",
  gaussian = FALSE
)
# If dependent variable has more than 2 levels, consider fct_lump:
mtcars |> dplyr::mutate(gear=factor(gear) |> forcats::fct_lump_n(n=1)) |>
compare2numvars(dep_vars="wt",indep_var="gear",gaussian=TRUE)
```

---

compare2qualvars *Comparison for columns of factors for 2 groups*

Description

compare2qualvars computes fisher.test with simulated p-value and descriptive statistics for a group of categorical dependent variables.

Usage

```
compare2qualvars(
  data,
  dep_vars,
  indep_var,
  round_p = 3,
  round_desc = 2,
  pretext = FALSE,
  mark = FALSE,
  singleline = FALSE,
  spacer = " ",
  linebreak = "\n",
  p_subgroups = FALSE
)
```

Arguments

data	name of data set (tibble/data.frame) to analyze.
dep_vars	vector of column names for dependent variables.
indep_var	name of grouping variable, has to translate to 2 groups.
round_p	level for rounding p-value.
round_desc	number of significant digits for rounding of descriptive stats.

`pretext`	for function formatP.
`mark`	for function formatP.
`singleline`	Put all group levels in a single line?
`spacer`	Text element to indent levels and fill empty cells, defaults to " ".
`linebreak`	place holder for newline.
`p_subgroups`	test subgroups by recoding other levels into other, default is not to do this.

Value

A tibble with variable names, descriptive statistics, and p-value, number of rows is number of dep_vars.

Examples

```
compare2qualvars(
  data = mtcars, dep_vars = c("gear", "cyl", "carb"), indep_var = "am",
  spacer = " "
)
compare2qualvars(
  data = mtcars, dep_vars = c("gear", "cyl", "carb"), indep_var = "am",
  spacer = " ", singleline = TRUE
)
compare2qualvars(
  data = mtcars, dep_vars = c("gear", "cyl", "carb"), indep_var = "am",
  spacer = " ", p_subgroups = TRUE
)
```

`compare_n_numvars` *Comparison for columns of Gaussian or ordinal measures for n groups*

Description

Some names were changed in August 2022, to reflect the update of the function to handle ordinal data using non-parametric equivalents.

Usage

```
compare_n_numvars(
  .data = rawdata,
  dep_vars,
  indep_var,
  gaussian,
  round_desc = 2,
  range = FALSE,
  rangesep = " ",
  pretext = FALSE,
  mark = FALSE,
  round_p = 3,
  add_n = FALSE
)
```

Arguments

`.data`	name of dataset (tibble/data.frame) to analyze, defaults to rawdata.
`dep_vars`	vector of column names.
`indep_var`	name of grouping variable.
`gaussian`	Logical specifying normal or ordinal indep_var (and chooses comparison tests accordingly)
`round_desc`	number of significant digits for rounding of descriptive stats.
`range`	include min/max?
`rangesep`	text between statistics and range or other elements.
`pretext, mark`	for function formatP.
`round_p`	level for rounding p-value.
`add_n`	add n to descriptive statistics?

Value

A list with elements "results": tibble with descriptive statistics, p-value from ANOVA/Kruskal-Wallis test, p-values for pairwise comparisons, significance indicators, and descriptives pasted with significance. "raw": nested list with output from all underlying analyses.

Examples

```
# Usually,only the result table is relevant:
compare_n_numvars(
  .data = mtcars, dep_vars = c("wt", "mpg", "hp"),
  indep_var = "drat",
  gaussian = TRUE
)$results
# For a report, result columns may be filtered as needed:
compare_n_numvars(
  .data = mtcars, dep_vars = c("wt", "mpg", "hp"),
  indep_var = "cyl",
  gaussian = FALSE
)$results |>
  dplyr::select(Variable, `cyl 4 fn`:`cyl 8 fn`, multivar_p)
```

`compare_n_qualvars` *Comparison for columns of factors for more than 2 groups with post-hoc*

Description

Comparison for columns of factors for more than 2 groups with post-hoc

Usage

```
compare_n_qualvars(
  data,
  dep_vars,
  indep_var,
  round_p = 3,
  round_desc = 2,
  pretext = FALSE,
  mark = FALSE,
  singleline = FALSE,
  spacer = " ",
  linebreak = "\n",
  prettynum = FALSE
)
```

Arguments

`data`	name of data set (tibble/data.frame) to analyze.
`dep_vars`	vector of column names.
`indep_var`	name of grouping variable.
`round_p`	level for rounding p-value.
`round_desc`	number of significant digits for rounding of descriptive stats
`pretext`	for function formatP
`mark`	for function formatP
`singleline`	Put all group levels in a single line?
`spacer`	Text element to indent levels, defaults to " ".
`linebreak`	place holder for newline.
`prettynum`	Apply prettyNum to results?

Value

A tibble with variable names, descriptive statistics, and p-value of fisher.test and pairwise_fisher_test, number of rows is number of dep_vars.

Examples

```
# Separate lines for each factor level:
compare_n_qualvars(
  data = mtcars, dep_vars = c("am", "cyl", "carb"), indep_var = "gear",
  spacer = " "
)
# All levels in one row but with linebreaks:
compare_n_qualvars(
  data = mtcars, dep_vars = c("am", "cyl", "carb"), indep_var = "gear",
  singleline = TRUE
)
# All levels in one row, separateted by ";":
compare_n_qualvars(
  data = mtcars, dep_vars = c("am", "cyl", "carb"), indep_var = "gear",
  singleline = TRUE, linebreak = "; "
)
```

---

cortestR *Correlations with significance*

Description

`cortestR` computes correlations and their significance level based on cor.test. Coefficients and p-values may be combined or reported separately.

Usage

```
cortestR(
  cordata,
  method = "pearson",
  digits = 3,
  digits_p = 3,
  sign_symbol = TRUE,
  split = FALSE,
  space = ""
)
```

Arguments

`cordata`	data frame or matrix with rawdata.
`method`	as in cor.test.
`digits`	rounding level for estimate.
`digits_p`	rounding level for p value.
`sign_symbol`	If true, use significance indicator instead of p-value.
`split`	logical, report correlation and p combined (default) or split in list.
`space`	character to fill empty upper triangle.

Value

Depending on parameters split and sign_symbol, either a single data frame with coefficient and p-values or significance symbols or a list with two data frames.

Examples

```
# with defaults
cortestR(mtcars[, c("wt", "mpg", "qsec")], split = FALSE, sign_symbol = TRUE)
# separate coefficients and p-values
cortestR(mtcars[, c("wt", "mpg", "qsec")], split = TRUE, sign_symbol = FALSE)
```

`detect_outliers` *Find outliers based on IQR*

Description

[Experimental]

`detect_outliers` computes IQR and finds outliers. It gives the same results as `geom_boxplot` and thus differs slightly from `boxplot.stats`.

Usage

```
detect_outliers(x, coef = 1.5)
```

Arguments

`x`	numeric vector.
`coef`	coefficient for boxplot.stats, defaults to 1.5.

Value

A list with elements positions and outliers as numeric vectors.

Examples

```
detect_outliers(rnorm(100))
```

eGFR — *Estimation of glomerular filtration rate (eGFR) based on sex, age, and either serum creatinine and/or cystatin C*

Description

[Experimental]

`eGFR` computes eGFR according to different rules (see references).

Usage

```
eGFR(data, age_var = "age", sex_var = "sex", crea_var = NULL, cys_var = NULL)
```

Arguments

`data`	name of data set (tibble/data.frame) to analyze.
`age_var`	name of column with patient age in years, default=age.
`sex_var`	name of column with sex, assumed as female and male.
`crea_var`	name of column with creatinine in mg/dl. If not available, leave as NULL.
`cys_var`	name of column with cystatin C in mg/l. If not available, leave as NULL.

Value

A list with 3 elements:

eGFR_crea

eGFR_cystatin

eGFR_creatinine_cystatin

References

https://www.kidney.org/content/ckd-epi-creatinine-cystatin-equation-2021

https://www.kidney.org/content/ckd-epi-creatinine-equation-2021

https://www.kidney.org/content/ckd-epi-cystatin-c-equation-2012

`faketrial` *Results from a simulated clinical trial with interaction effects.*

Description

A dataset containing physiological data, biomarkers, and categorical data.

Usage

```
faketrial
```

Format

A tibble with 300 rows and 24 variables:

Sex Sex of animal, factor with levels 'female', 'male'

Agegroup Factor with levels 'young','middle','old'

Treatment Factor with levels 'sham', 'OP'

HR Heart rate

sysRR,diaRR Systolic and diastolic blood pressure

Med xxx Pseudo-medications, factors with levels 'y','n'

Biomarker x units Biomarkers with log-normal distribution

Responder factor yes/no, systolic plood pressure >= 120?

`FindVars` *Find numeric index and names of columns based on patterns*

Description

[Superseded]

Function ColSeeker extends this by adding class-checks.

`FindVars` looks up colnames (by default for data-frame rawdata) based on parts of names, using regular expressions. Be warned that special characters as e.g. `[ (` need to be escaped or replaced by `.` Exclusion rules may be specified as well. New function `ColSeeker()` extends this by adding class-checks.

Usage

```
FindVars(
  varnames,
  allnames = colnames(rawdata),
  exact = FALSE,
  exclude = NA,
  casesensitive = TRUE,
  fixed = FALSE,
  return_symbols = FALSE
)
```

Arguments

`varnames`	Vector of pattern to look for.
`allnames`	Vector of values to detect pattern in; by default: colnames(rawdata).
`exact`	Partial matching or exact only (adding ^ and $)?
`exclude`	Vector of pattern to exclude from found names.
`casesensitive`	Logical if case is respected in matching (default FALSE: a<>A)
`fixed`	Logical, match as is, argument is passed to `grep()`.
`return_symbols`	Should names be reported as symbols additionally? (Default FALSE)

Value

A list with index, names, backticked names, and symbols

Examples

```
FindVars(varnames = c("^c", "g"), allnames = colnames(mtcars))
FindVars(varnames = c("^c", "g"), allnames = colnames(mtcars), exclude = "r")
```

`flex2rmd` *Transform flextable to rmd if non-interactive*

Description

`flex2rmd` takes a flextable and returns a markdown table if not in an interactive session

Usage

```
flex2rmd(ft)
```

Arguments

`ft`	a flextable

Value

either a markdown table or the flextable

formatP — *Re-format p-values, avoiding rounding to 0 and adding surprisal if requested*

Description

`formatP` simplifies p-values by rounding to the maximum of p or a predefined level. Optionally < or = can be added, as well as symbols according to significance level.

Usage

```
formatP(
  pIn,
  ndigits = 3,
  textout = TRUE,
  pretext = FALSE,
  mark = FALSE,
  german_num = FALSE,
  add.surprisal = FALSE,
  sprecision = 1
)
```

Arguments

`pIn`	A numeric vector or matrix with p-values.
`ndigits`	Number of digits (default=3).
`textout`	Cast output to character (default=TRUE)?
`pretext`	Should = or < be added before p (default=FALSE)?
`mark`	Should significance level be added after p (default=FALSE)?
`german_num`	change dot (default) to comma?
`add.surprisal`	Add surprisal aka Shannon information to p-value (default=FALSE)?
`sprecision`	Rounding level for surprisal (default=1).

Value

vector or matrix (depending on type of pIn) with type character (default) or numeric, depending on parameter textout

Examples

```
formatP(0.012345)
formatP(0.012345, add.surprisal = TRUE)
formatP(0.012345, ndigits = 4)
formatP(0.000122345, ndigits = 3, pretext = TRUE)
```

`ggcormat` *Print graphical representation of a correlation matrix.*

Description

`ggcormat` makes the same correlation matrix as cortestR and graphically represents it in a plot

Usage

```
ggcormat(
  cor_mat,
  p_mat = NULL,
  method = "Correlation",
  title = "",
  maxpoint = 2.1,
  textsize = 5,
  axistextsize = 2,
  titlesize = 3,
  breaklabels = NULL,
  lower_only = TRUE,
  .low = "blue3",
  .high = "red2",
  .legendtitle = NULL
)
```

Arguments

`cor_mat`	correlation matrix as produced by cor.
`p_mat`	Optional matrix of p-values; if provided, this is used to define size of dots rather than absolute correlation.
`method`	text specifying type of correlation.
`title`	plot title.
`maxpoint`	maximum for scale_size_manual, may need adjustment depending on plotsize.
`textsize`	for theme text.
`axistextsize`	relative text size for axes.
`titlesize`	as you already guessed, relative text size for title.
`breaklabels`	currently not used, intended for str_wrap.
`lower_only`	should only lower triangle be plotted?
`.low`	Color for heatmap.
`.high`	Color for heatmap.
`.legendtitle`	Optional name for color legend.

Value

A ggplot object, allowing further styling.

Examples

```
coeff_pvalues <- cortestR(mtcars[, c("wt", "mpg", "qsec", "hp")],
  split = TRUE, sign_symbol = FALSE
)
# focus on coefficients:
ggcormat(cor_mat = coeff_pvalues$corout, maxpoint = 5)
# size taken from p-value:
ggcormat(
  cor_mat = coeff_pvalues$corout,
  p_mat = coeff_pvalues$pout, maxpoint = 5)
```

`glmCI` *Confidence interval for generalized linear models*

Description

`glm_CI` computes and formats CIs for glm.

Usage

```
glmCI(model, min = .01, max = 100, cisep = '\U000022ef', ndigit=2)
```

Arguments

`model`	Output from glm.
`min, max`	Lower and upper limits for CIs, useful for extremely wide CIs.
`cisep`	Separator between CI values.
`ndigit`	rounding level.

Value

A list with coefficient, CIs, and pasted coef([CIs]).

Examples

```
glm_out <- glm(am ~ mpg, family = binomial, data = mtcars)
glmCI(glm_out)
```

ksnormal *Kolmogorov-Smirnov-Test against Normal distribution*

Description

`ksnormal` is a convenience function around ks.test, testing against Normal distribution. If less than 2 values are provided, NA is returned.

Usage

```
ksnormal(x)
```

Arguments

`x`	Vector of data to test.

Value

p.value from ks.test.

Examples

```
# original ks.test:
ks.test(
  x = mtcars$wt, pnorm, mean = mean(mtcars$wt, na.rm = TRUE),
  sd = sd(mtcars$wt, na.rm = TRUE)
)
# wrapped version:
ksnormal(x = mtcars$wt)
```

label_outliers *Add labels to outliers in boxplot/beeswarm.*

Description

[Experimental]

`label_outliers` adds a text_repel layer to an existing ggplot object. It is intended to be used with boxplots or beeswarm plots. Faceting will result in separate computations for outliers. It requires the `ggrepel` package.

Usage

```
label_outliers(
  plotbase,
  labelvar = NULL,
  coef = 1.5,
  nudge_x = 0,
  nudge_y = 0,
  color = "darkred",
  size = 3,
  hjust = 0,
  face = "bold"
)
```

Arguments

`plotbase`	ggplot object to add labels to.
`labelvar`	variable to use as label. If NULL, rownames or rownumbers are used.
`coef`	coefficient for boxplot.stats, defaults to 1.5.
`nudge_x`	nudge in x direction, defaults to 0.
`nudge_y`	nudge in y direction, defaults to 0.
`color`	color of labels, defaults to darkred.
`size`	size of labels, defaults to 3.
`hjust`	horizontal justification of labels, defaults to 0.
`face`	font face of labels, defaults to bold.

Value

A ggplot object, allowing further styling.

`logrange_1` *Predefined sets of labels for plots with log-scaled axes*

Description

`logrange_1` returns a vector for log-labels at .1, 1, 100, 1000 ...

Usage

```
logrange_1

logrange_5

logrange_123456789

logrange_12357

logrange_15
```

Format

An object of class `numeric` of length 41.

An object of class `numeric` of length 738.

An object of class `numeric` of length 369.

An object of class `numeric` of length 205.

An object of class `numeric` of length 82.

Value

numeric vector

numeric vector

Functions

- `logrange_5`: vector for log-labels at 1.0, 1.5, 2.0, 2.5 ... 10, 15, 20, 25 ...
- `logrange_123456789`: vector for log-labels at 1, 2, 3 ... 9, 10, 20, 30 ... 90, 100 ...
- `logrange_12357`: vector for log-labels at 1 ,2, 3, 5, 7, 10, 20 ,30, 50, 70 ...
- `logrange_15`: vector for log-labels at 1, 5, 10, 50 ...

Examples

```
ggplot2::ggplot(mtcars) +
ggplot2::aes(wt, mpg) +
  ggplot2::geom_point() +
  ggplot2::scale_y_log10(breaks = lograngе_5)
ggplot2::ggplot(mtcars) +
  ggplot2::aes(wt, mpg) +
  ggplot2::geom_point() +
  ggplot2::scale_y_log10(breaks = lograngе_123456789)
```

`markSign` *Convert significance levels to symbols*

Description

`markSign` returns the symbol associated with a significance level.

Usage

```
markSign(SignIn, plabel = c("n.s.", "+", "*", "**", "***"))
```

Arguments

`SignIn`	A single p-value.
`plabel`	A translation table, predefined with the usual symbols.

Value

factor with label as defined in plabel.

Examples

```
markSign(0.012)
```

`meansd`	*Compute mean and sd and put together with the ± symbol.*

Description

Compute mean and sd and put together with the ± symbol.

Usage

```
meansd(
  x,
  roundDig = 2,
  drop0 = FALSE,
  groupvar = NULL,
  range = FALSE,
  rangesep = " ",
  add_n = FALSE,
  .german = FALSE
)
```

Arguments

`x`	Data for computation.
`roundDig`	Number of relevant digits for roundR.
`drop0`	Should trailing zeros be dropped?
`groupvar`	Optional grouping variable for subgroups.
`range`	Should min and max be included in output?
`rangesep`	How should min/max be separated from mean+-sd?
`add_n`	Should n be included in output?
`.german`	logical, should "." and "," be used as bigmark and decimal?

Value

character vector with mean ± SD, rounded to desired precision

Examples

```
# basic usage of meansd
meansd(x = mtcars$wt)
# with additional options
meansd(x = mtcars$wt, groupvar = mtcars$am, add_n = TRUE)
```

meanse — *Compute mean and standard error of mean and put together with the ± symbol.*

Description

`meanse` computes SEM based on Standard Deviation/square root(n)

Usage

```
meanse(x, mult = 1, roundDig = 2, drop0 = FALSE)
```

Arguments

`x`	Data for computation.
`mult`	multiplier for SEM, default 1, can be set to e.g. 2 or 1.96 to create confidence intervals
`roundDig`	Number of relevant digits for roundR.
`drop0`	Should trailing zeros be dropped?

Value

character vector with mean ± SEM, rounded to desired precision

Examples

```
# basic usage of meanse
meanse(x = mtcars$wt)
```

`medianse` *Compute standard error of median.*

Description

`medianse` is based on `mad`/square root(n)

Usage

```
medianse(x)
```

Arguments

`x`	Data for computation.

Value

numeric vector with SE Median.

Examples

```
# basic usage of medianse
medianse(x = mtcars$wt)
```

`median_cl_boot` *Compute confidence interval of median by bootstrapping.*

Description

`median_cl_boot` computes lower and upper confidence limits for the estimated median, based on bootstrapping.

Usage

```
median_cl_boot(x, conf = 0.95, type = "basic", nrepl = 10^3)
```

Arguments

`x`	Data for computation.
`conf`	confidence interval with default 95%.
`type`	type for function boot.ci.
`nrepl`	number of bootstrap replications, defaults to 1000.

Value

A tibble with one row and three columns: Median, CIlow, CIhigh.

Examples

```
# basic usage of median_cl_boot
median_cl_boot(x = mtcars$wt)
```

---

median_cl_boot_gg *Rename output from median_cl_boot for use in ggplot.*

Description

median_cl_boot_gg computes lower and upper confidence limits for the estimated median, based on bootstrapping, using default settings.

Usage

```
median_cl_boot_gg(x)
```

Arguments

x	Data for computation.

Value

A tibble with one row and three columns: y, ymin, ymax.

Examples

```
# basic usage of median_cl_boot
median_cl_boot_gg(x = mtcars$wt)
```

---

median_quart *Compute median and quartiles and put together.*

Description

Compute median and quartiles and put together.

Usage

```
median_quart(
  x,
  nround = NULL,
  probs = c(0.25, 0.5, 0.75),
  qtype = 8,
  roundDig = 2,
  drop0 = FALSE,
  groupvar = NULL,
  range = FALSE,
  rangesep = " ",
  rangearrow = " -> ",
  prettynum = FALSE,
  .german = FALSE,
  add_n = FALSE
)
```

Arguments

`x`	Data for computation.
`nround`	Number of digits for fixed round.
`probs`	Quantiles to compute.
`qtype`	Type of quantiles.
`roundDig`	Number of relevant digits for roundR.
`drop0`	Should trailing zeros be dropped?
`groupvar`	Optional grouping variable for subgroups.
`range`	Should min and max be included in output?
`rangesep`	How should min/max be separated from mean+-sd?
`rangearrow`	What is put between min -> max?
`prettynum`	logical, apply prettyNum to results?
`.german`	logical, should "." and "," be used as bigmark and decimal?
`add_n`	Should n be included in output?

Value

character vector with median `[1stQuartile/3rdQuartile]`, rounded to desired precision

Examples

```
# basic usage of median_quart
median_quart(x = mtcars$wt)
# with additional options
median_quart(x = mtcars$wt, groupvar = mtcars$am, add_n = TRUE)
data(faketrial)
median_quart(x=faketrial$`Biomarker 1 [units]`,groupvar = faketrial$Treatment)
```

pairwise_fisher_test *Pairwise Fisher's exact tests*

Description

pairwise_fisher_test calculates pairwise comparisons between group levels with corrections for multiple testing.

Usage

```
pairwise_fisher_test(
  dep_var,
  indep_var,
  adjmethod = "fdr",
  plevel = 0.05,
  symbols = letters[-1],
  ref = FALSE
)
```

Arguments

dep_var	dependent variable, containing the data.
indep_var	independent variable, should be factor or coercible.
adjmethod	method for adjusting p values (see p.adjust).
plevel	threshold for significance.
symbols	predefined as b,c, d...; provides footnotes to mark group differences, e.g. b means different from group 2
ref	is the 1st subgroup the reference (like in Dunnett test)?

Value

A list with elements "methods" (character), "p.value" (matrix), "plevel" (numeric), and "sign_colwise" (vector of length number of levels - 1)

Examples

```
# All pairwise comparisons
pairwise_fisher_test(dep_var = mtcars$cyl, indep_var = mtcars$gear)
# Only comparison against reference gear=3
pairwise_fisher_test(dep_var = mtcars$cyl, indep_var = mtcars$gear, ref = TRUE)
```

`pairwise_ordcat_test` *Pairwise comparison for ordinal categories*

Description

`pairwise_ordcat_test` calculates pairwise comparisons for ordinal categories between all group levels with corrections for multiple testing.

Usage

```
pairwise_ordcat_test(
  dep_var,
  indep_var,
  adjmethod = "fdr",
  plevel = 0.05,
  symbols = letters[-1],
  ref = FALSE,
  cmh = TRUE
)
```

Arguments

`dep_var`	dependent variable, containing the data
`indep_var`	independent variable, should be factor
`adjmethod`	method for adjusting p values (see p.adjust)
`plevel`	threshold for significance
`symbols`	predefined as b,c, d...; provides footnotes to mark group differences, e.g. b means different from group 2
`ref`	is the 1st subgroup the reference (like in Dunnett test)
`cmh`	Should Cochran-Mantel-Haenszel test (cmh_test) be used for testing? If false, the linear-by-linear association test (lbl_test) is applied.

Value

A list with elements "methods" (character), "p.value" (matrix), "plevel" (numeric), and "sign_colwise" (vector of length number of levels - 1)

Examples

```
# All pairwise comparisons
mtcars2 <- dplyr::mutate(mtcars, cyl = factor(cyl, ordered = TRUE))
pairwise_ordcat_test(dep_var = mtcars2$cyl, indep_var = mtcars2$gear)
# Only comparison against reference gear=3
pairwise_ordcat_test(dep_var = mtcars2$cyl, indep_var = mtcars2$gear, ref = TRUE)
```

`pairwise_t_test` *Extended pairwise t-test*

Description

`pairwise_t_test`calculate pairwise comparisons between group levels with corrections for multiple testing based on pairwise.t.test

Usage

```
pairwise_t_test(
  dep_var,
  indep_var,
  adjmethod = "fdr",
  plevel = 0.05,
  symbols = letters[-1]
)
```

Arguments

`dep_var`	dependent variable, containing the data
`indep_var`	independent variable, should be factor
`adjmethod`	method for adjusting p values (see p.adjust)
`plevel`	threshold for significance
`symbols`	predefined as b,c, d...; provides footnotes to mark group differences, e.g. b means different from group 2

Value

A list with method output of pairwise.t.test, matrix of p-values, and character vector with significance indicators.

Examples

```
pairwise_t_test(dep_var = mtcars$wt, indep_var = mtcars$cyl)
```

`pairwise_wilcox_test` *Pairwise Wilcoxon tests*

Description

`pairwise_wilcox_test` calculates pairwise comparisons on ordinal data between all group levels with corrections for multiple testing based on wilcox_test from package 'coin'.

Usage

```
pairwise_wilcox_test(
  dep_var,
  indep_var,
  strat_var = NA,
  adjmethod = "fdr",
  distr = "exact",
  plevel = 0.05,
  symbols = letters[-1],
  sep = ""
)
```

Arguments

`dep_var`	dependent variable, containing the data.
`indep_var`	independent variable, should be factor.
`strat_var`	optional factor for stratification.
`adjmethod`	method for adjusting p values (see p.adjust)
`distr`	Computation of p-values, see wilcox_test.
`plevel`	threshold for significance.
`symbols`	predefined as b,c, d...; provides footnotes to mark group differences, e.g. b means different from group 2.
`sep`	text between statistics and range or other elements.

Value

A list with matrix of adjusted p-values and character vector with significance indicators.

Examples

```
pairwise_wilcox_test(dep_var = mtcars$wt, indep_var = mtcars$cyl)
```

pdf_kable *Enhanced kable with latex*

Description

pdf_kable formats tibbles/df's for markdown

Usage

```
pdf_kable(
  .input,
  width1 = 6,
  twidth = 14,
  tposition = "left",
  innercaption = NULL,
  caption = "",
  foot = NULL,
  escape = TRUE
)
```

Arguments

.input	table to print
width1	Width of 1st column, default 6.
twidth	Default 14
tposition	Default left
innercaption	subheader
caption	header
foot	footnote
escape	see kable

Value

A character vector of the table source code.

plot_LB *Lineweaver-Burk diagram*

Description

`plot_LB` plots a Lineweaver-Burk diagram and computes the linear model

Usage

```
plot_LB(
  data,
  substrate,
  velocity,
  group = NULL,
  title = "Lineweaver-Burk-Plot",
  xlab = "1/substrate",
  ylab = "1/velocity"
)
```

Arguments

`data`	data structure with columns for model data
`substrate`	colname for substrate concentration
`velocity`	colname for reaction velocity
`group`	colname for optional grouping factor
`title`	title of the plot
`xlab`	label of the abscissa
`ylab`	label of the ordinate

Examples

```
MMdata <- data.frame(subst = c(2.00, 1.00, 0.50, 0.25),
                 velo = c(0.2253, 0.1795, 0.1380, 0.1000))

plot_LB(data=MMdata,
        substrate = 'subst',velocity = 'velo')

MMdata <- data.frame(subst = rep(c(2.00, 1.00, 0.50, 0.25),2),
                 velo = c(0.2253, 0.1795, 0.1380, 0.1000,
                          0.4731333, 0.4089333, 0.3473000, 0.2546667),
                 condition = rep(c('C1','C2'),each=4))

plot_LB(data=MMdata,substrate = 'subst',
        velocity = 'velo',group='condition')
```

plot_MM *Michaelis-Menten enzyme kinetics model and plot*

Description

plot_MM creates a Michaelis-Menten type Enzyme kinetics plot and returns model as well

Usage

```
plot_MM(
  data,
  substrate,
  velocity,
  group = NULL,
  title = "Michaelis-Menten",
  xlab = "substrate",
  ylab = "velocity"
)
```

Arguments

data	data structure with columns for model data
substrate	colname for substrate concentration
velocity	colname for reaction velocity
group	colname for optional grouping factor
title	title of the plot
xlab	label for x-axis
ylab	label for y-axis

Value

a list with elements "MMfit" and "MMplot"

Examples

```
MMdata <- data.frame(subst = c(2.00, 1.00, 0.50, 0.25),
               velo = c(0.2253, 0.1795, 0.1380, 0.1000))

plot_MM(data=MMdata,
       substrate = 'subst',velocity = 'velo')

MMdata <- data.frame(subst = rep(c(2.00, 1.00, 0.50, 0.25),2),
               velo = c(0.2253, 0.1795, 0.1380, 0.1000,
                        0.4731333, 0.4089333, 0.3473000, 0.2546667),
               condition = rep(c('C1','C2'),each=4))
```

```
plot_MM(data=MMdata,substrate = 'subst',
        velocity = 'velo',group='condition')
```

`print_kable` *Enhanced kable with definable number of rows and/or columns for splitting*

Description

[Superseded]

package flextable is a more powerful alternative

`print_kable` formats and prints tibbles/df's in markdown with splitting into sub-tables with repeated caption and header.

Usage

```
print_kable(t, nrows = 30, caption = "", ncols = 100, ...)
```

Arguments

`t`	table to print.
`nrows`	number of rows (30) before splitting.
`caption`	header.
`ncols`	number of columns (100) before splitting.
`...`	Further arguments passed to kable.

Value

No return value, called for side effects.

Examples

```
## Not run:
print_kable(mtcars, caption = "test")

## End(Not run)
```

roundR — *Automatic rounding to a reasonable length, based on largest number*

Description

roundR takes a vector or matrix of numbers and returns rounded values with selected precision and various formatting options.

Usage

```
roundR(
  roundin,
  level = 2,
  smooth = FALSE,
  textout = TRUE,
  drop0 = FALSE,
  .german = FALSE,
  .bigmark = FALSE
)
```

Arguments

roundin	A vector or matrix of numbers.
level	A number specifying number of relevant digits to keep.
smooth	A logical specifying if you want rounding before the dot (e.g. 12345 to 12300).
textout	A logical if output is converted to text.
drop0	A logical if trailing zeros should be dropped.
.german	A logical if german numbers should be reported.
.bigmark	A logical if big.mark is to be shown, mark itself depends on parameter .german.

Value

vector of type character (default) or numeric, depending on parameter textout.

Examples

```
roundR(1.23456, level = 3)
roundR(1.23456, level = 3, .german = TRUE)
roundR(1234.56, level = 2, smooth = TRUE)
```

SEM *Standard Error of Mean.*

Description

SEM computes standard error of mean.

Usage

```
SEM(x)
```

Arguments

x	Data for computation.

Value

numeric vector with SEM.

Examples

```
SEM(x = mtcars$wt)
```

se_median *Compute standard error of median*

Description

se_median is based on mad/square root(n) (Deprecated, please see medianse, which is the same but named more consistently)

Usage

```
se_median(x)
```

Arguments

x	Data for computation.

Value

numeric vector with SE Median.

Examples

```
# basic usage of se_median
## Not run:
se_median(x = mtcars$wt)

## End(Not run)
```

`surprisal` *Compute surprisal aka Shannon information from p-values*

Description

`surprisal` takes p-values and returns s, a value representing the number of consecutive heads on a fair coin, that would be as surprising as the p-value

Usage

```
surprisal(p, precision = 1)
```

Arguments

`p`	a vector of p-values
`precision`	rounding level with default 1

Value

a character vector of s-values

`tab.search` *Search within data.frame or tibble*

Description

`tab.search` searches for pattern within a data-frame or tibble, returning column(s) and row(s)

Usage

```
tab.search(searchdata = rawdata, pattern, find.all = T, names.only = FALSE)
```

Arguments

`searchdata`	table to search in, predefined as rawdata
`pattern`	regex, for exact matches add ^findme$
`find.all`	return all row indices or only 1st per column,default=TRUE
`names.only`	return only vector of colnames rather than list with names and rows, default=FALSE

Value

A list with numeric vectors for each column giving row numbers of matched elements

---

`t_var_test` *Independent sample t-test with test for equal variance*

Description

`t_var_test` tests for equal variance based on var.test and calls t.test, setting the option var.equal accordingly.

Usage

```
t_var_test(data, formula, cutoff = 0.05)
```

Arguments

`data`	Tibble or data_frame.
`formula`	Formula object with dependent and independent variable.
`cutoff`	is significance threshold for equal variances.

Value

A list from t.test

Examples

```
t_var_test(mtcars, wt ~ am)
# may be used in pipes:
mtcars |> t_var_test(wt ~ am)
```

`var_coeff` *Compute coefficient of variance.*

Description

`var_coeff computes relative variability as standard deviation/mean *100`

Usage

```
var_coeff(x)
```

Arguments

`x`	Data for computation.

Value

numeric vector with coefficient of variance.

Examples

```
var_coeff(x = mtcars$wt)
```

WINratio — *Comparison for groups in clinical trials based on all possible combinations of subjects*

Description

[Experimental]

`WINratio` computes the ratio of wins and losses for any number of comparison rules.

Usage

```
WINratio(data, groupvar, testvars, rules, idvar = NULL, p_digits = 3)
```

Arguments

`data`	name of data set (tibble/data.frame) to analyze.
`groupvar`	name of grouping variable, has to translate to 2 groups.
`testvars`	names of variables for sequential rules.
`rules`	list of rules (minimal cut-offs) for sequential comparison, negative if reduction is success, positive if increase is beneficial, must not be 0.
`idvar`	name of identifier variable. If NULL, rownumber is used.
`p_digits`	level for rounding p-value.

Value

A list with elements:

WINratio=vector with WINratio and CIs,

WINodds=odds ratio of wins and losses, taking ties into account,

p.value=p.value from prop.test,

WINratioCI=character with merged WINratio, CI, and p

testdata= tibble with testdata from cross-join.

Index