Package 'statsExpressions'

October 9, 2026

Type Package

Title Tidy Dataframes and Expressions with Statistical Details

Version 2.1.2

Maintainer Indrajeet Patil <patilindrajeet.science@gmail.com>

Description Utilities for producing dataframes with rich details for the most common types of statistical approaches and tests: parametric, nonparametric, robust, and Bayesian t-test, one-way ANOVA, correlation analyses, contingency table analyses, and meta-analyses. The functions are pipe-friendly and provide a consistent syntax to work with tidy data. These dataframes additionally contain expressions with statistical details, and can be used in graphing packages. This package also forms the statistical processing backend for 'ggstatsplot'. References: Patil (2021) <doi:10.21105/joss.03236>.

License MIT + file LICENSE

URL https://www.indrapatil.com/statsExpressions/, https://github.com/IndrajeetPatil/statsExpressions

BugReports https://github.com/IndrajeetPatil/statsExpressions/issues

Depends R (>= 4.5.0), stats

Imports afex (>= 1.5-1), BayesFactor (>= 0.9.12-4.8), bayestestR (>= 0.19.0), correlation (>= 0.8.8), datawizard (>= 1.4.0), dplyr (>= 1.2.1), effectsize (>= 1.0.3), glue (>= 1.8.1), insight (>= 1.5.4), parameters (>= 0.29.3), performance (>= 0.18.2), PMCMRplus (>= 1.9.12), purrr (>= 1.2.2), rlang (>= 1.3.0), rstantools (>= 2.7.1), tidyr (>= 1.3.2), withr (>= 3.0.3), WRS2 (>= 1.1-7)

Suggests ggplot2 (>= 4.0.3), knitr (>= 1.52), metaBMA (>= 0.6.9), metafor (>= 5.2-1), metaplus (>= 1.0-10), patrick (>= 0.3.1), rmarkdown (>= 2.32), survival, testthat (>= 3.3.2), utils

VignetteBuilder knitr

Config/Needs/check anthonynorth/roxyglobals, any::patrick

Config/roxygen2/version 8.1.1

Config/roxyglobals/unique TRUE

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

LazyData true

NeedsCompilation no

Author Indrajeet Patil [cre, aut, cph] (ORCID:
<https://orcid.org/0000-0003-1995-6531>)

Repository CRAN

Date/Publication 2026-10-09 10:10:02 UTC

Contents

add_expression_col 2
bugs_long 4
centrality_description 5
contingency_table 8
corr_test 14
extract_stats_type 17
iris_long 18
long_to_wide_converter 19
meta_analysis 20
movies_long 24
oneway_anova 25
one_sample_test 30
pairwise_comparisons 34
pairwise_contingency_table 40
tidy_model_expressions 43
tidy_model_parameters 45
two_sample_test 46

Index **52**

`add_expression_col` *Template for expressions with statistical details*

Description

Creates an expression from a data frame containing statistical details. Ideally, this data frame would come from having run `tidy_model_parameters()` on your model object.

This function is currently **not** stable and should not be used outside of this package context.

Usage

```
add_expression_col(
  data,
  paired = FALSE,
  statistic.text = NULL,
  effsize.text = NULL,
  prior.type = NULL,
  n = NULL,
  n.text = ifelse(paired, list(quote(italic("n")["pairs"])),
    list(quote(italic("n")["obs"]))),
  digits = 2L,
  digits.df = 0L,
  digits.df.error = digits.df,
  ...
)
```

Arguments

`data`	A data frame containing details from the statistical analysis and should contain some or all of the following columns: • *statistic*: the numeric value of a statistic. • *df.error*: the numeric value of a parameter being modeled (often degrees of freedom for the test); irrelevant if there are no degrees of freedom. • *df*: relevant if the statistic in question has two degrees of freedom. • *p.value*: the *p*-value associated with the observed statistic. • *method*: method describing the test carried out. • *effectsize*: name of the effect size (if not present, same as `method`). • *estimate*: estimated value of the effect size. • *conf.level*: width for the confidence intervals. • *conf.low*: lower bound for effect size estimate. • *conf.high*: upper bound for effect size estimate. • *bf10*: Bayes Factor value. If this column is present, the analysis is treated as Bayesian and a Bayesian expression template is used, which additionally needs the *conf.method* and *prior.scale* columns.
`paired`	Logical that decides whether the experimental design is repeated measures/within-subjects or between-subjects. The default is `FALSE`.
`statistic.text`	A list containing a `language` object that specifies the relevant test statistic. For example, for tests with *t*-statistic, `statistic.text = list(quote(italic("t")))`. If `NULL` (default), it is inferred from the `method` column.
`effsize.text`	A list containing a `language` object that specifies the relevant effect size or posterior estimate. For example, `effsize.text = list(quote(italic("d")))`. If `NULL` (default), it is inferred from the `effectsize` column.
`prior.type`	Currently ignored. The prior label is inferred from the `method` column.
`n`	An integer specifying the sample size used for the test. Ignored if `data` already contains an `n.obs` column.

`n.text` A list containing a `language` object that specifies the design, which will determine what the `n` stands for. It defaults to `quote(italic("n")["pairs"])` if `paired = TRUE`, and to `quote(italic("n")["obs"])` if `paired = FALSE`.

`digits`, `digits.df`, `digits.df.error`
Number of decimal places to display for the parameters (`digits`; default: `2L`), and for the degrees of freedom in the `df` (`digits.df`; default: `0L`) and `df.error` (`digits.df.error`; default: same as `digits.df`) columns.

`...` Currently ignored.

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

Examples

```
set.seed(123)

# creating a data frame with stats results
stats_df <- cbind.data.frame(
  statistic  = 5.494,
  df         = 29.234,
  p.value    = 0.00001,
  estimate   = -1.980,
  conf.level = 0.95,
  conf.low   = -2.873,
  conf.high  = -1.088,
  method     = "Student's t-test"
)

# expression for *t*-statistic with Cohen's *d* as effect size
# note that the plotmath expressions need to be quoted
add_expression_col(
  data           = stats_df,
  statistic.text = list(quote(italic("t"))),
  effsize.text   = list(quote(italic("d"))),
  n              = 32L,
  n.text         = list(quote(italic("n")["no.obs"])),
  digits         = 3L,
  digits.df      = 3L
)
```

---

bugs_long *Tidy version of the "Bugs" dataset.*

Description

Tidy version of the "Bugs" dataset.

Usage

```
bugs_long
```

Format

A data frame with 372 rows and 6 variables

- subject. Dummy identity number for each participant.
- gender. Participant's gender (Female, Male).
- region. Region of the world the participant was from.
- education. Level of education.
- condition. Condition of the experiment the participant gave rating for (**LDLF**: low disgustingness and low frighteningness; **LDHF**: low disgustingness and high frighteningness; **HDLF**: high disgustingness and low frighteningness; **HDHF**: high disgustingness and high frighteningness).
- desire. The desire to kill an arthropod was indicated on a scale from 0 to 10.

Details

This data set, "Bugs", provides the extent to which men and women want to kill arthropods that vary in frighteningness (low, high) and disgustingness (low, high). Each participant rates their attitudes towards all arthropods. Subset of the data reported by Ryan et al. (2013).

References

Ryan, R. S., Wilde, M., & Crist, S. (2013). Compared to a small, supervised lab experiment, a large, unsupervised web-based experiment on a previously unknown effect has benefits that outweigh its potential costs. *Computers in Human Behavior*, *29*(4), 1295-1301.

Examples

```
dim(bugs_long)
head(bugs_long)
dplyr::glimpse(bugs_long)
```

---

centrality_description

Data frame and expression for distribution properties

Description

Parametric, non-parametric, robust, and Bayesian measures of centrality.

Usage

```
centrality_description(
  data,
  x,
  y,
  type = "parametric",
  conf.level = 0.95,
  tr = 0.2,
  digits = 2L,
  ...
)
```

Arguments

`data`	A data frame (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix, table, array, etc.) will **not** be accepted. Additionally, grouped data frames from `{dplyr}` should be ungrouped before they are entered as `data`.
`x`	The grouping (or independent) variable in `data`.
`y`	The response (or outcome or dependent) variable from `data`.
`type`	A character specifying the type of statistical approach: • `"parametric"` • `"nonparametric"` • `"robust"` • `"bayes"` You can specify just the initial letter (e.g. `"np"` or `"bf"`). Matching is on the initial lowercase letter only, so any other value (including capitalized values such as `"Bayes"`) falls back to `"parametric"` without a warning.
`conf.level`	Scalar between `0` and `1` (default: `95%` confidence/credible intervals, `0.95`).
`tr`	Trim level for the mean when carrying out `robust` tests. In case of an error, try reducing the value of `tr`, which is by default set to `0.2`. Lowering the value might help.
`digits`	Number of digits for rounding or significant figures. May also be `"signif"` to return significant figures or `"scientific"` to return scientific notation. Control the number of digits by adding the value as suffix, e.g. `digits = "scientific4"` to have scientific notation with 4 decimal places, or `digits = "signif5"` for 5 significant figures (see also `signif()`).
`...`	Currently ignored.

Details

This function describes a distribution for `y` variable for each level of the grouping variable in `x` by a set of indices (e.g., measures of centrality, dispersion, range, skewness, kurtosis, etc.). It additionally returns an expression containing a specified centrality measure. The function internally relies on `datawizard::describe_distribution()` function.

Value

The returned object is a tibble data frame with the additional class `"statsExpressions"`. Unlike the hypothesis-testing functions, this one returns one row **per level** of the grouping variable `x`, describing the distribution of `y` within that level. The columns are:

- the grouping variable (named after `x`, e.g. `Species`)
- the requested centrality estimate (named after `y`, e.g. `Sepal.Length`): the mean (`type = "parametric"`), median (`"nonparametric"`), trimmed mean (`"robust"`), or MAP estimate (`"bayes"`)
- `std.dev`: standard deviation (for `"parametric"` and `"robust"`)
- `mad`: median absolute deviation (for `"nonparametric"`)
- `iqr`: interquartile range
- `conf.low`, `conf.high`: bounds of the confidence/credible interval for the centrality estimate
- `min`, `max`: minimum and maximum values
- `skewness`, `kurtosis`: shape statistics of the distribution
- `n.obs`: number of non-missing observations
- `missing.obs`: number of missing observations
- `expression`: a list-column of pre-formatted plotmath expressions (each element is a `language` object, not a character string) containing the centrality estimate, ready to be used in `{ggplot2}`
- `n.expression`: a label combining the group name and its sample size (e.g. `"setosa\n(n = 50)"`), useful as an axis or facet label

The exact dispersion column depends on the `type` of analysis (`std.dev` vs `mad`; neither is returned for `"bayes"`), and `datawizard::describe_distribution()` is used internally to compute these indices.

For more examples, see the data frame output vignette and the Return value schema article.

Centrality measures

The table below provides summary about:

- statistical test carried out for inferential statistics
- type of effect size estimate and a measure of uncertainty for this estimate
- functions used internally to compute these details

Type	Measure	Function used
Parametric	mean	`datawizard::describe_distribution()`
Non-parametric	median	`datawizard::describe_distribution()`
Robust	trimmed mean	`datawizard::describe_distribution()`
Bayesian	MAP	`datawizard::describe_distribution()`

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

Examples

```
# for reproducibility
set.seed(123)

# ---------------------- parametric ----------------------

centrality_description(iris, Species, Sepal.Length, type = "parametric")

# ---------------------- non-parametric ------------------

centrality_description(mtcars, am, wt, type = "nonparametric")

# ---------------------- robust --------------------------

centrality_description(ToothGrowth, supp, len, type = "robust")

# ---------------------- Bayesian ------------------------

centrality_description(sleep, group, extra, type = "bayes")
```

---

contingency_table *Contingency table analyses*

Description

Parametric and Bayesian one-way and two-way contingency table analyses.

Usage

```
contingency_table(
  data,
  x,
  y = NULL,
  paired = FALSE,
  type = "parametric",
  counts = NULL,
  ratio = NULL,
  alternative = "two.sided",
  digits = 2L,
  conf.level = 0.95,
  sampling.plan = "indepMulti",
  fixed.margin = "rows",
  prior.concentration = 1,
  ...
)
```

Arguments

`data`	A data frame (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix, table, array, etc.) will **not** be accepted. Additionally, grouped data frames from `{dplyr}` should be ungrouped before they are entered as `data`.
`x`	The variable to use as the **rows** in the contingency table.
`y`	The variable to use as the **columns** in the contingency table. Default is `NULL`. If `NULL`, one-sample proportion test (a goodness of fit test) will be run for the `x` variable.
`paired`	Logical indicating whether data came from a within-subjects or repeated measures design study (Default: `FALSE`). Only relevant for two-way tables (i.e., when `y` is supplied), and paired designs are only supported for frequentist tests (McNemar's test). For a two-way table with `type = "bayes"`, `paired` is ignored and the Bayesian test of independence for unpaired data is run instead. For one-way tables (`y = NULL`), `paired` has no effect.
`type`	A character specifying the type of statistical approach: • `"parametric"` • `"nonparametric"` • `"robust"` • `"bayes"` You can specify just the initial letter (e.g. `"np"` or `"bf"`). Matching is on the initial lowercase letter only, so any other value (including capitalized values such as `"Bayes"`) falls back to `"parametric"` without a warning.
`counts`	The variable in data containing counts, or `NULL` if each row represents a single observation.
`ratio`	A vector of proportions: the expected proportions for the proportion test (should sum to 1). Default is `NULL`, which means the null is equal theoretical proportions across the levels of the nominal variable. E.g., `ratio = c(0.5, 0.5)` for two levels, `ratio = c(0.25, 0.25, 0.25, 0.25)` for four levels, etc.
`alternative`	A character string specifying the alternative hypothesis; Controls the type of CI returned: `"two.sided"` (default, two-sided CI), `"greater"` or `"less"` (one-sided CI). Partial matching is allowed (e.g., `"g"`, `"l"`, `"two"`...). See section *One-Sided CIs* in the effectsize_CIs vignette.
`digits`	Number of digits for rounding or significant figures. May also be `"signif"` to return significant figures or `"scientific"` to return scientific notation. Control the number of digits by adding the value as suffix, e.g. `digits = "scientific4"` to have scientific notation with 4 decimal places, or `digits = "signif5"` for 5 significant figures (see also `signif()`).
`conf.level`	Scalar between `0` and `1` (default: `95%` confidence/credible intervals, `0.95`).
`sampling.plan`	Character describing the sampling plan. Possible options: • `"indepMulti"` (independent multinomial; default) • `"poisson"` • `"jointMulti"` (joint multinomial)

- "hypergeom" (hypergeometric).

Only used for Bayesian two-way tables. For more, see BayesFactor::contingencyTableBF().

fixed.margin — For the independent multinomial sampling plan, which margin is fixed ("rows" or "cols"). Defaults to "rows". Only used for Bayesian two-way tables.

prior.concentration — Specifies the prior concentration parameter, set to 1 by default. It indexes the expected deviation from the null hypothesis under the alternative, and corresponds to Gunel and Dickey's (1974) "a" parameter. Only used for Bayesian analyses.

... — Additional arguments (currently ignored).

Value

The returned object is a tibble data frame with the additional class "statsExpressions". The exact set of columns depends on the test and, for functions that accept a type argument, on the chosen analysis (parametric, non-parametric, robust, or Bayesian). Any given call therefore returns *some* (not all) of the columns below.

Hypothesis testing

- statistic: the numeric value of a statistic
- df: the numeric value of a parameter being modeled (often degrees of freedom for the test)
- df.error and df: relevant only if the statistic in question has two degrees of freedom (e.g. anova)
- p.value: the *p*-value associated with the observed statistic (two-sided unless a one-sided alternative is requested, where supported)
- method: the name of the inferential statistical test

Effect size estimation

- effectsize: the name of the effect size
- estimate: estimated value of the effect size
- conf.level: the coverage level of the confidence/credible interval (e.g. 0.95); the interval itself spans conf.low to conf.high
- conf.low: lower bound for the effect size estimate
- conf.high: upper bound for the effect size estimate
- conf.method: method used to compute the confidence/credible interval
- conf.distribution: statistical distribution for the effect

Bayesian analysis (only when type = "bayes")

- bf10: Bayes factor for the alternative hypothesis relative to the null
- log_e_bf10: natural logarithm of the Bayes factor (present for most, but not all, Bayesian analyses)
- prior.distribution, prior.scale, prior.location: prior specification used to compute the Bayes factor and posterior estimates

Pairwise comparisons (for pairwise_comparisons() and pairwise_contingency_table())

- `group1`, `group2`: the two levels being compared
- `p.adjust.method`: the adjustment method used for multiple comparisons
- `p.value.adj`: the adjusted *p*-value; returned by `pairwise_contingency_table()`. Note that `pairwise_comparisons()` instead folds the adjusted value into `p.value` (and does not return a separate `p.value.adj` column)

Common columns

- `n.obs`: number of observations
- `expression`: a list-column of pre-formatted plotmath expressions; each element is a `language` object (not a character string) containing the statistical details, ready to be used in `{ggplot2}` (e.g. in `labs()` or `annotate()`)

For a per-function, column-by-column breakdown of the output (and an explanation of the internal `add_expression_col()` engine that builds the `expression` column), see the Return value schema article. For more examples, see the data frame output vignette.

Contingency table analyses

The table below provides summary about:

- statistical test carried out for inferential statistics
- type of effect size estimate and a measure of uncertainty for this estimate
- functions used internally to compute these details

There are no dedicated non-parametric or robust contingency table analyses. `type = "nonparametric"` and `type = "robust"` are accepted, but run the same frequentist tests as `type = "parametric"` (the "Parametric/Non-parametric" rows below).

two-way table:
Hypothesis testing

Type	Design	Test	Function used
Parametric/Non-parametric	Unpaired	Pearson's chi-squared test	`stats::chisq.test()`
Bayesian	Unpaired	Bayesian Pearson's chi-squared test	`BayesFactor::contingencyTableBF()`
Parametric/Non-parametric	Paired	McNemar's chi-squared test	`stats::mcnemar.test()`
Bayesian	Paired	No	No

Effect size estimation

Type	Design	Effect size	CI available?	Function used
Parametric/Non-parametric	Unpaired	Cramer's *V*	Yes	`effectsize::cramers_v()`
Bayesian	Unpaired	Cramer's *V*	Yes	`effectsize::cramers_v()`
Parametric/Non-parametric	Paired	Cohen's *g*	Yes	`effectsize::cohens_g()`
Bayesian	Paired	No	No	No

Paired Bayesian analysis is not supported: for a two-way table with `type = "bayes"`, the `paired` argument is ignored and the unpaired Bayesian test is run instead.

one-way table:

Hypothesis testing

Type	Test	Function used
Parametric/Non-parametric	Goodness of fit chi-squared test	`stats::chisq.test()`
Bayesian	Bayesian Goodness of fit chi-squared test	(custom)

Effect size estimation

Type	Effect size	CI available?	Function used
Parametric/Non-parametric	Pearson's *C*	Yes	`effectsize::pearsons_c()`
Bayesian	No	No	No

Examples

```
#### -------------------- association test ----------------------- ####

# ------------------------ frequentist --------------------------------

# unpaired

set.seed(123)
contingency_table(
  data = mtcars,
  x = am,
  y = vs,
  paired = FALSE
)

# paired

paired_data <- dplyr::tibble(
  response_before = structure(
    c(1L, 2L, 1L, 2L),
    levels = c("no", "yes"),
    class = "factor"
  ),
  response_after = structure(
    c(1L, 1L, 2L, 2L),
    levels = c("no", "yes"),
    class = "factor"
  ),
  Freq = c(65L, 25L, 5L, 5L)
)

set.seed(123)
contingency_table(
  data = paired_data,
  x = response_before,
  y = response_after,
```

```
  paired = TRUE,
  counts = Freq
)

# ------------------------ Bayesian -------------------------------------

# unpaired

set.seed(123)
contingency_table(
  data = mtcars,
  x = am,
  y = vs,
  paired = FALSE,
  type = "bayes"
)

# paired

set.seed(123)
contingency_table(
  data = paired_data,
  x = response_before,
  y = response_after,
  paired = TRUE,
  counts = Freq,
  type = "bayes"
)

#### -------------------- goodness-of-fit test -------------------- ####

# ------------------------ frequentist ----------------------------------

set.seed(123)
contingency_table(
  data = as.data.frame(HairEyeColor),
  x = Eye,
  counts = Freq
)

# ------------------------ Bayesian -------------------------------------

set.seed(123)
contingency_table(
  data = as.data.frame(HairEyeColor),
  x = Eye,
  counts = Freq,
  ratio = c(0.2, 0.2, 0.3, 0.3),
  type = "bayes"
)
```

`corr_test` *Correlation analyses*

Description

Parametric, non-parametric, robust, and Bayesian correlation test.

Usage

```
corr_test(
  data,
  x,
  y,
  type = "parametric",
  digits = 2L,
  conf.level = 0.95,
  tr = 0.2,
  bf.prior = 0.707,
  ...
)
```

Arguments

`data`	A data frame (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix, table, array, etc.) will **not** be accepted. Additionally, grouped data frames from `{dplyr}` should be ungrouped before they are entered as `data`.
`x`	The column in `data` containing the explanatory variable to be plotted on the `x`-axis.
`y`	The column in `data` containing the response (outcome) variable to be plotted on the `y`-axis.
`type`	A character specifying the type of statistical approach: • `"parametric"` • `"nonparametric"` • `"robust"` • `"bayes"` You can specify just the initial letter (e.g. `"np"` or `"bf"`). Matching is on the initial lowercase letter only, so any other value (including capitalized values such as `"Bayes"`) falls back to `"parametric"` without a warning.
`digits`	Number of digits for rounding or significant figures. May also be `"signif"` to return significant figures or `"scientific"` to return scientific notation. Control the number of digits by adding the value as suffix, e.g. `digits = "scientific4"` to have scientific notation with 4 decimal places, or `digits = "signif5"` for 5 significant figures (see also `signif()`).

conf.level	Scalar between 0 and 1 (default: 95% confidence/credible intervals, 0.95).
tr	Trim level for the mean when carrying out robust tests. In case of an error, try reducing the value of tr, which is by default set to 0.2. Lowering the value might help.
bf.prior	A number between 0.5 and 2 (default 0.707), the prior width to use in calculating Bayes factors and posterior estimates. In addition to numeric arguments, several named values are also recognized: "medium", "wide", and "ultrawide", corresponding to *r* scale values of 1/2, sqrt(2)/2, and 1, respectively. In case of an ANOVA, this value corresponds to scale for fixed effects.
...	Additional arguments (currently ignored).

Value

The returned object is a tibble data frame with the additional class "statsExpressions". The exact set of columns depends on the test and, for functions that accept a type argument, on the chosen analysis (parametric, non-parametric, robust, or Bayesian). Any given call therefore returns *some* (not all) of the columns below.

Hypothesis testing

- statistic: the numeric value of a statistic
- df: the numeric value of a parameter being modeled (often degrees of freedom for the test)
- df.error and df: relevant only if the statistic in question has two degrees of freedom (e.g. anova)
- p.value: the *p*-value associated with the observed statistic (two-sided unless a one-sided alternative is requested, where supported)
- method: the name of the inferential statistical test

Effect size estimation

- effectsize: the name of the effect size
- estimate: estimated value of the effect size
- conf.level: the coverage level of the confidence/credible interval (e.g. 0.95); the interval itself spans conf.low to conf.high
- conf.low: lower bound for the effect size estimate
- conf.high: upper bound for the effect size estimate
- conf.method: method used to compute the confidence/credible interval
- conf.distribution: statistical distribution for the effect

Bayesian analysis (only when type = "bayes")

- bf10: Bayes factor for the alternative hypothesis relative to the null
- log_e_bf10: natural logarithm of the Bayes factor (present for most, but not all, Bayesian analyses)
- prior.distribution, prior.scale, prior.location: prior specification used to compute the Bayes factor and posterior estimates

Pairwise comparisons (for `pairwise_comparisons()` and `pairwise_contingency_table()`)

- `group1`, `group2`: the two levels being compared
- `p.adjust.method`: the adjustment method used for multiple comparisons
- `p.value.adj`: the adjusted *p*-value; returned by `pairwise_contingency_table()`. Note that `pairwise_comparisons()` instead folds the adjusted value into `p.value` (and does not return a separate `p.value.adj` column)

Common columns

- `n.obs`: number of observations
- `expression`: a list-column of pre-formatted plotmath expressions; each element is a `language` object (not a character string) containing the statistical details, ready to be used in `{ggplot2}` (e.g. in `labs()` or `annotate()`)

For a per-function, column-by-column breakdown of the output (and an explanation of the internal `add_expression_col()` engine that builds the `expression` column), see the Return value schema article. For more examples, see the data frame output vignette.

Correlation analyses

The table below provides summary about:

- statistical test carried out for inferential statistics
- type of effect size estimate and a measure of uncertainty for this estimate
- functions used internally to compute these details

Hypothesis testing and **Effect size estimation**

Type	Test	CI available?	Function used
Parametric	Pearson's correlation coefficient	Yes	`correlation::correlation()`
Non-parametric	Spearman's rank correlation coefficient	Yes	`correlation::correlation()`
Robust	Winsorized Pearson's correlation coefficient	Yes	`correlation::correlation()`
Bayesian	Bayesian Pearson's correlation coefficient	Yes	`correlation::correlation()`

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

Examples

```
# for reproducibility
set.seed(123)

# ---------------------- parametric ----------------------

corr_test(mtcars, wt, mpg, type = "parametric")
```

```
# ----------------------- non-parametric -------------------

corr_test(mtcars, wt, mpg, type = "nonparametric")

# ----------------------- robust ---------------------------

corr_test(mtcars, wt, mpg, type = "robust")

# ----------------------- Bayesian -------------------------

corr_test(mtcars, wt, mpg, type = "bayes")
```

`extract_stats_type` *Switch the type of statistics.*

Description

Relevant mostly for {ggstatsplot} and {statsExpressions} packages, where different statistical approaches are supported via this argument: parametric, non-parametric, robust, and Bayesian. This switch function converts strings entered by users to a common pattern for convenience.

`stats_type_switch()` is an alias of `extract_stats_type()`.

Usage

```
extract_stats_type(type)

stats_type_switch(type)
```

Arguments

type	A character specifying the type of statistical approach: • "parametric" • "nonparametric" • "robust" • "bayes" You can specify just the initial letter (e.g. "np" or "bf"). Matching is on the initial lowercase letter only, so any other value (including capitalized values such as "Bayes") falls back to "parametric" without a warning.

Value

A character vector of the same length as type, with values "parametric", "nonparametric", "robust", or "bayes".

Examples

```
extract_stats_type("p")
extract_stats_type("bf")
extract_stats_type(c("np", "robust", "Bayes"))
```

`iris_long` *Edgar Anderson's Iris Data in long format.*

Description

Edgar Anderson's Iris Data in long format.

Usage

```
iris_long
```

Format

A data frame with 600 rows and 6 variables

- id. Dummy identity number for each flower (150 flowers in total).
- Species. The species are *Iris setosa*, *versicolor*, and *virginica*.
- condition. Factor giving a detailed description of the attribute (Four levels: `"Petal.Length"`, `"Petal.Width"`, `"Sepal.Length"`, `"Sepal.Width"`).
- attribute. What attribute is being measured (`"Sepal"` or `"Petal"`).
- measure. What aspect of the attribute is being measured (`"Length"` or `"Width"`).
- value. Value of the measurement.

Details

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

This is a modified dataset from `{datasets}` package.

Examples

```
dim(iris_long)
head(iris_long)
dplyr::glimpse(iris_long)
```

`long_to_wide_converter` *Convert long/tidy data frame to wide format*

Description

This conversion is helpful mostly for repeated measures design, where removing `NA`s by participant can be a bit tedious.

Usage

```
long_to_wide_converter(
  data,
  x,
  y,
  subject.id = NULL,
  paired = TRUE,
  spread = TRUE,
  ...
)
```

Arguments

`data`	A data frame (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix, table, array, etc.) will **not** be accepted. Additionally, grouped data frames from `{dplyr}` should be ungrouped before they are entered as `data`.
`x`	The grouping (or independent) variable from `data`. For repeated measures designs, see the note on ordering in `subject.id`.
`y`	The response (or outcome or dependent) variable from `data`.
`subject.id`	Relevant in case of a repeated measures or within-subjects design (i.e., `paired = TRUE`), it specifies the subject or repeated measures identifier. **Important**: If this argument is `NULL` (which is the default), observations are paired by their row order within each level of `x` (i.e., the data is assumed to be sorted in a subject-1, subject-2, ... pattern within every level). If the data is **not** sorted this way, the paired results will be silently incorrect, so it is safest to always specify `subject.id`.
`paired`	Logical that decides whether the experimental design is repeated measures/within-subjects or between-subjects. The default is `FALSE`.
`spread`	Logical that decides whether the data frame needs to be converted from long/tidy to wide (default: `TRUE`).
`...`	Currently ignored.

Value

A tibble with NAs removed while respecting the between-or-within-subjects nature of the dataset: for paired designs, a subject with a missing value in any condition is removed entirely, while for unpaired designs only the rows with missing values are removed. Rows are grouped by `subject.id` whenever it is supplied, so with `paired = FALSE` and a `subject.id`, a missing value still removes every row of that subject. The `.rowid` column contains the subject identifier (or an internal row identifier).

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

Examples

```
# for reproducibility
library(statsExpressions)
set.seed(123)

# repeated measures design
long_to_wide_converter(
  bugs_long,
  condition,
  desire,
  subject.id = subject,
  paired = TRUE
)

# independent measures design
long_to_wide_converter(mtcars, cyl, wt, paired = FALSE)
```

---

meta_analysis *Random-effects meta-analysis*

Description

Parametric, robust, and Bayesian random-effects meta-analysis.

A non-parametric meta-analysis is not available, so `type = "nonparametric"` is not supported.

Usage

```
meta_analysis(
  data,
  type = "parametric",
  random = "mixture",
  digits = 2L,
  conf.level = 0.95,
```

```
  ...
)
```

Arguments

`data`	A data frame. It **must** contain columns named `estimate` (effect sizes or outcomes) and `std.error` (corresponding standard errors). These two columns will be used: • as `yi` and `sei` arguments in `metafor::rma()` (for **parametric** test) • as `yi` and `sei` arguments in `metaplus::metaplus()` (for **robust** test) • as `y` and `SE` arguments in `metaBMA::meta_random()` (for **Bayesian** test)
`type`	A character specifying the type of statistical approach: • `"parametric"` • `"nonparametric"` • `"robust"` • `"bayes"` You can specify just the initial letter (e.g. `"np"` or `"bf"`). Matching is on the initial lowercase letter only, so any other value (including capitalized values such as `"Bayes"`) falls back to `"parametric"` without a warning.
`random`	The type of random-effects distribution for the robust meta-analysis: `"mixture"` (default; mixture of normals), `"normal"`, or `"t-dist"` (*t*-distribution). Passed to `metaplus::metaplus()` and only used when `type = "robust"`.
`digits`	Number of digits for rounding or significant figures. May also be `"signif"` to return significant figures or `"scientific"` to return scientific notation. Control the number of digits by adding the value as suffix, e.g. `digits = "scientific4"` to have scientific notation with 4 decimal places, or `digits = "signif5"` for 5 significant figures (see also `signif()`).
`conf.level`	Scalar between `0` and `1` (default: `95%` confidence/credible intervals, `0.95`).
`...`	Additional arguments passed to the respective meta-analysis function.

Value

The returned object is a tibble data frame with the additional class `"statsExpressions"`. The exact set of columns depends on the test and, for functions that accept a `type` argument, on the chosen analysis (parametric, non-parametric, robust, or Bayesian). Any given call therefore returns *some* (not all) of the columns below.

Hypothesis testing

- `statistic`: the numeric value of a statistic
- `df`: the numeric value of a parameter being modeled (often degrees of freedom for the test)
- `df.error` and `df`: relevant only if the statistic in question has two degrees of freedom (e.g. anova)
- `p.value`: the *p*-value associated with the observed statistic (two-sided unless a one-sided `alternative` is requested, where supported)
- `method`: the name of the inferential statistical test

Effect size estimation

- `effectsize`: the name of the effect size
- `estimate`: estimated value of the effect size
- `conf.level`: the coverage level of the confidence/credible interval (e.g. `0.95`); the interval itself spans `conf.low` to `conf.high`
- `conf.low`: lower bound for the effect size estimate
- `conf.high`: upper bound for the effect size estimate
- `conf.method`: method used to compute the confidence/credible interval
- `conf.distribution`: statistical distribution for the effect

Bayesian analysis (only when `type = "bayes"`)

- `bf10`: Bayes factor for the alternative hypothesis relative to the null
- `log_e_bf10`: natural logarithm of the Bayes factor (present for most, but not all, Bayesian analyses)
- `prior.distribution`, `prior.scale`, `prior.location`: prior specification used to compute the Bayes factor and posterior estimates

Pairwise comparisons (for `pairwise_comparisons()` and `pairwise_contingency_table()`)

- `group1`, `group2`: the two levels being compared
- `p.adjust.method`: the adjustment method used for multiple comparisons
- `p.value.adj`: the adjusted *p*-value; returned by `pairwise_contingency_table()`. Note that `pairwise_comparisons()` instead folds the adjusted value into `p.value` (and does not return a separate `p.value.adj` column)

Common columns

- `n.obs`: number of observations
- `expression`: a list-column of pre-formatted plotmath expressions; each element is a `language` object (not a character string) containing the statistical details, ready to be used in `{ggplot2}` (e.g. in `labs()` or `annotate()`)

For a per-function, column-by-column breakdown of the output (and an explanation of the internal `add_expression_col()` engine that builds the `expression` column), see the Return value schema article. For more examples, see the data frame output vignette.

Random-effects meta-analysis

The table below provides summary about:

- statistical test carried out for inferential statistics
- type of effect size estimate and a measure of uncertainty for this estimate
- functions used internally to compute these details

Hypothesis testing and **Effect size estimation**

Type	Test	Effect size	CI available?	Function used
Parametric	Meta-analysis via random-effects models	*beta*	Yes	`metafor::rma()`
Robust	Meta-analysis via robust random-effects models	*beta*	Yes	`metaplus::metaplus()`
Bayesian	Meta-analysis via Bayesian random-effects models	*beta*	Yes	`metaBMA::meta_random()`

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

Note

Important: The function assumes that you have already downloaded the needed package (`{metafor}`, `{metaplus}`, or `{metaBMA}`) for meta-analysis. If they are not available, you will be asked to install them.

Examples

```
set.seed(123)
library(statsExpressions)

# let's use `mag` dataset from `{metaplus}`
data(mag, package = "metaplus")
dat <- dplyr::rename(mag, estimate = yi, std.error = sei)

# ---------------------- parametric ----------------------------------------



meta_analysis(dat)



# ---------------------- robust ----------------------------------------------

meta_analysis(dat, type = "robust", random = "normal")



# ---------------------- Bayesian --------------------------------------------

meta_analysis(dat, type = "bayes")
```

`movies_long` *Movie information and user ratings from IMDB.*

Description

Movie information and user ratings from IMDB.

Usage

```
movies_long
```

Format

A data frame with 1,579 rows and 8 variables

- title. Title of the movie.
- year. Year of release.
- budget. Total budget (if known) in US dollars
- length. Length in minutes.
- rating. Average IMDB user rating.
- votes. Number of IMDB users who rated this movie.
- mpaa. MPAA rating.
- genre. Genre of the movie (`"Action"`, `"Action Comedy"`, `"Action Drama"`, `"Animated"`, `"Comedy"`, `"Comedy Drama"`, `"Drama"`, `"Romance Drama"`, `"RomCom"`).

Details

Modified dataset from `{ggplot2movies}` package.

Source

https://CRAN.R-project.org/package=ggplot2movies

Examples

```
dim(movies_long)
head(movies_long)
dplyr::glimpse(movies_long)
```

oneway_anova *One-way analysis of variance (ANOVA)*

Description

Parametric, non-parametric, robust, and Bayesian one-way ANOVA.

Usage

```
oneway_anova(
  data,
  x,
  y,
  subject.id = NULL,
  type = "parametric",
  paired = FALSE,
  digits = 2L,
  conf.level = 0.95,
  effsize.type = "omega",
  var.equal = FALSE,
  bf.prior = 0.707,
  tr = 0.2,
  nboot = 100L,
  ...
)
```

Arguments

data	A data frame (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix, table, array, etc.) will **not** be accepted. Additionally, grouped data frames from {dplyr} should be ungrouped before they are entered as data.
x	The grouping (or independent) variable from data. For repeated measures designs, see the note on ordering in subject.id.
y	The response (or outcome or dependent) variable from data.
subject.id	Relevant in case of a repeated measures or within-subjects design (i.e., paired = TRUE), it specifies the subject or repeated measures identifier. **Important**: If this argument is NULL (which is the default), observations are paired by their row order within each level of x (i.e., the data is assumed to be sorted in a subject-1, subject-2, ... pattern within every level). If the data is **not** sorted this way, the paired results will be silently incorrect, so it is safest to always specify subject.id.
type	A character specifying the type of statistical approach: • "parametric" • "nonparametric"

- "robust"
- "bayes"

You can specify just the initial letter (e.g. "np" or "bf"). Matching is on the initial lowercase letter only, so any other value (including capitalized values such as "Bayes") falls back to "parametric" without a warning.

paired	Logical that decides whether the experimental design is repeated measures/within-subjects or between-subjects. The default is FALSE.
digits	Number of digits for rounding or significant figures. May also be "signif" to return significant figures or "scientific" to return scientific notation. Control the number of digits by adding the value as suffix, e.g. digits = "scientific4" to have scientific notation with 4 decimal places, or digits = "signif5" for 5 significant figures (see also signif()).
conf.level	Scalar between 0 and 1 (default: 95% confidence/credible intervals, 0.95).
effsize.type	Type of effect size needed for *parametric* tests. The argument can be "eta" (partial eta-squared) or "omega" (partial omega-squared).
var.equal	a logical variable indicating whether to treat the two variances as being equal. If TRUE then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.
bf.prior	A number between 0.5 and 2 (default 0.707), the prior width to use in calculating Bayes factors and posterior estimates. In addition to numeric arguments, several named values are also recognized: "medium", "wide", and "ultrawide", corresponding to *r* scale values of 1/2, sqrt(2)/2, and 1, respectively. In case of an ANOVA, this value corresponds to scale for fixed effects.
tr	Trim level for the mean when carrying out robust tests. In case of an error, try reducing the value of tr, which is by default set to 0.2. Lowering the value might help.
nboot	Number of bootstrap samples for computing confidence interval for the effect size (Default: 100L).
...	Additional arguments (currently ignored).

Value

The returned object is a tibble data frame with the additional class "statsExpressions". The exact set of columns depends on the test and, for functions that accept a type argument, on the chosen analysis (parametric, non-parametric, robust, or Bayesian). Any given call therefore returns *some* (not all) of the columns below.

Hypothesis testing

- statistic: the numeric value of a statistic
- df: the numeric value of a parameter being modeled (often degrees of freedom for the test)
- df.error and df: relevant only if the statistic in question has two degrees of freedom (e.g. anova)
- p.value: the *p*-value associated with the observed statistic (two-sided unless a one-sided alternative is requested, where supported)

- `method`: the name of the inferential statistical test

Effect size estimation

- `effectsize`: the name of the effect size
- `estimate`: estimated value of the effect size
- `conf.level`: the coverage level of the confidence/credible interval (e.g. `0.95`); the interval itself spans `conf.low` to `conf.high`
- `conf.low`: lower bound for the effect size estimate
- `conf.high`: upper bound for the effect size estimate
- `conf.method`: method used to compute the confidence/credible interval
- `conf.distribution`: statistical distribution for the effect

Bayesian analysis (only when `type = "bayes"`)

- `bf10`: Bayes factor for the alternative hypothesis relative to the null
- `log_e_bf10`: natural logarithm of the Bayes factor (present for most, but not all, Bayesian analyses)
- `prior.distribution`, `prior.scale`, `prior.location`: prior specification used to compute the Bayes factor and posterior estimates

Pairwise comparisons (for `pairwise_comparisons()` and `pairwise_contingency_table()`)

- `group1`, `group2`: the two levels being compared
- `p.adjust.method`: the adjustment method used for multiple comparisons
- `p.value.adj`: the adjusted *p*-value; returned by `pairwise_contingency_table()`. Note that `pairwise_comparisons()` instead folds the adjusted value into `p.value` (and does not return a separate `p.value.adj` column)

Common columns

- `n.obs`: number of observations
- `expression`: a list-column of pre-formatted plotmath expressions; each element is a `language` object (not a character string) containing the statistical details, ready to be used in `{ggplot2}` (e.g. in `labs()` or `annotate()`)

For a per-function, column-by-column breakdown of the output (and an explanation of the internal `add_expression_col()` engine that builds the `expression` column), see the Return value schema article. For more examples, see the data frame output vignette.

One-way ANOVA

The table below provides summary about:

- statistical test carried out for inferential statistics
- type of effect size estimate and a measure of uncertainty for this estimate
- functions used internally to compute these details

between-subjects:

Hypothesis testing

Type	No. of groups	Test	Function used
Parametric	> 2	Fisher's or Welch's one-way ANOVA	`stats::oneway.test()`
Non-parametric	> 2	Kruskal-Wallis one-way ANOVA	`stats::kruskal.test()`
Robust	> 2	Heteroscedastic one-way ANOVA for trimmed means	`WRS2::t1way()`
Bayesian	> 2	Fisher's ANOVA	`BayesFactor::anovaBF()`

Effect size estimation

Type	No. of groups	Effect size	CI available?	Function used
Parametric	> 2	partial eta-squared, partial omega-squared	Yes	`effectsize::omega_square`
Non-parametric	> 2	rank epsilon squared	Yes	`effectsize::rank_epsilon`
Robust	> 2	Explanatory measure of effect size	Yes	`WRS2::t1way()`
Bayesian	> 2	Bayesian R-squared	Yes	`performance::r2_bayes()`

within-subjects:

Data requirement: Repeated measures tests assume a *complete* design with exactly **one observation per subject per condition**. If your data has multiple trials per cell, aggregate first (e.g., take the mean). Verify with `table(data$subject, data$condition)` — every cell should equal 1.

Hypothesis testing

Type	No. of groups	Test	Function used
Parametric	> 2	One-way repeated measures ANOVA	`afex::aov_ez`
Non-parametric	> 2	Friedman rank sum test	`stats::fried`
Robust	> 2	Heteroscedastic one-way repeated measures ANOVA for trimmed means	`WRS2::rmanov`
Bayesian	> 2	One-way repeated measures ANOVA	`BayesFactor:`

Effect size estimation

Type	No. of groups	Effect size	CI available?	Funct
Parametric	> 2	partial eta-squared, partial omega-squared	Yes	`effec`
Non-parametric	> 2	Kendall's coefficient of concordance	Yes	`effec`
Robust	> 2	Algina-Keselman-Penfield robust standardized difference average	Yes	`WRS2:`
Bayesian	> 2	Bayesian R-squared	Yes	`perfo`

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

Examples

```
# for reproducibility
set.seed(123)
library(statsExpressions)

# ---------------------- parametric -----------------------------------

# between-subjects
oneway_anova(
  data = mtcars,
  x    = cyl,
  y    = wt
)

# biased (partial eta-squared) effect size
oneway_anova(
  data         = mtcars,
  x            = cyl,
  y            = wt,
  effsize.type = "eta"
)

# within-subjects design
oneway_anova(
  data       = iris_long,
  x          = condition,
  y          = value,
  subject.id = id,
  paired     = TRUE
)

# ---------------------- non-parametric -------------------------------

# between-subjects
oneway_anova(
  data = mtcars,
  x    = cyl,
  y    = wt,
  type = "np"
)

# within-subjects design
oneway_anova(
  data       = iris_long,
  x          = condition,
  y          = value,
  subject.id = id,
  paired     = TRUE,
  type       = "np"
)

# ---------------------- robust ---------------------------------------
```

```
# between-subjects
oneway_anova(
  data = mtcars,
  x    = cyl,
  y    = wt,
  type = "r"
)

# within-subjects design
oneway_anova(
  data       = iris_long,
  x          = condition,
  y          = value,
  subject.id = id,
  paired     = TRUE,
  type       = "r"
)



# ---------------------- Bayesian -----------------------------------

# between-subjects
oneway_anova(
  data = mtcars,
  x    = cyl,
  y    = wt,
  type = "bayes"
)

# within-subjects design
oneway_anova(
  data       = iris_long,
  x          = condition,
  y          = value,
  subject.id = id,
  paired     = TRUE,
  type       = "bayes"
)
```

`one_sample_test` *One-sample tests*

Description

Parametric, non-parametric, robust, and Bayesian one-sample tests.

Usage

```
one_sample_test(
  data,
  x,
  type = "parametric",
  test.value = 0,
  alternative = "two.sided",
  digits = 2L,
  conf.level = 0.95,
  tr = 0.2,
  bf.prior = 0.707,
  effsize.type = "g",
  exact = FALSE,
  ...
)
```

Arguments

`data`	A data frame (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix, table, array, etc.) will **not** be accepted. Additionally, grouped data frames from `{dplyr}` should be ungrouped before they are entered as `data`.
`x`	A numeric variable from the data frame `data`.
`type`	A character specifying the type of statistical approach: • `"parametric"` • `"nonparametric"` • `"robust"` • `"bayes"` You can specify just the initial letter (e.g. `"np"` or `"bf"`). Matching is on the initial lowercase letter only, so any other value (including capitalized values such as `"Bayes"`) falls back to `"parametric"` without a warning.
`test.value`	A number indicating the true value of the mean (Default: `0`).
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.
`digits`	Number of digits for rounding or significant figures. May also be `"signif"` to return significant figures or `"scientific"` to return scientific notation. Control the number of digits by adding the value as suffix, e.g. `digits = "scientific4"` to have scientific notation with 4 decimal places, or `digits = "signif5"` for 5 significant figures (see also `signif()`).
`conf.level`	Scalar between `0` and `1` (default: `95%` confidence/credible intervals, `0.95`).
`tr`	Trim level for the mean when carrying out `robust` tests. In case of an error, try reducing the value of `tr`, which is by default set to `0.2`. Lowering the value might help.

bf.prior	A number between 0.5 and 2 (default 0.707), the prior width to use in calculating Bayes factors and posterior estimates. In addition to numeric arguments, several named values are also recognized: "medium", "wide", and "ultrawide", corresponding to *r* scale values of 1/2, sqrt(2)/2, and 1, respectively. In case of an ANOVA, this value corresponds to scale for fixed effects.
effsize.type	Type of effect size needed for *parametric* tests. The argument can be "d" (for Cohen's *d*) or "g" (for Hedge's *g*).
exact	A logical indicating whether you want exact p-values to be computed. Relevant only when type = "nonparametric" (Default: FALSE).
...	Currently ignored.

Value

The returned object is a tibble data frame with the additional class "statsExpressions". The exact set of columns depends on the test and, for functions that accept a type argument, on the chosen analysis (parametric, non-parametric, robust, or Bayesian). Any given call therefore returns *some* (not all) of the columns below.

Hypothesis testing

- statistic: the numeric value of a statistic
- df: the numeric value of a parameter being modeled (often degrees of freedom for the test)
- df.error and df: relevant only if the statistic in question has two degrees of freedom (e.g. anova)
- p.value: the *p*-value associated with the observed statistic (two-sided unless a one-sided alternative is requested, where supported)
- method: the name of the inferential statistical test

Effect size estimation

- effectsize: the name of the effect size
- estimate: estimated value of the effect size
- conf.level: the coverage level of the confidence/credible interval (e.g. 0.95); the interval itself spans conf.low to conf.high
- conf.low: lower bound for the effect size estimate
- conf.high: upper bound for the effect size estimate
- conf.method: method used to compute the confidence/credible interval
- conf.distribution: statistical distribution for the effect

Bayesian analysis (only when type = "bayes")

- bf10: Bayes factor for the alternative hypothesis relative to the null
- log_e_bf10: natural logarithm of the Bayes factor (present for most, but not all, Bayesian analyses)
- prior.distribution, prior.scale, prior.location: prior specification used to compute the Bayes factor and posterior estimates

Pairwise comparisons (for `pairwise_comparisons()` and `pairwise_contingency_table()`)

- `group1`, `group2`: the two levels being compared
- `p.adjust.method`: the adjustment method used for multiple comparisons
- `p.value.adj`: the adjusted *p*-value; returned by `pairwise_contingency_table()`. Note that `pairwise_comparisons()` instead folds the adjusted value into `p.value` (and does not return a separate `p.value.adj` column)

Common columns

- `n.obs`: number of observations
- `expression`: a list-column of pre-formatted plotmath expressions; each element is a `language` object (not a character string) containing the statistical details, ready to be used in `{ggplot2}` (e.g. in `labs()` or `annotate()`)

For a per-function, column-by-column breakdown of the output (and an explanation of the internal `add_expression_col()` engine that builds the `expression` column), see the Return value schema article. For more examples, see the data frame output vignette.

One-sample tests

The table below provides summary about:

- statistical test carried out for inferential statistics
- type of effect size estimate and a measure of uncertainty for this estimate
- functions used internally to compute these details

Hypothesis testing

Type	Test	Function used
Parametric	One-sample Student's *t*-test	`stats::t.test()`
Non-parametric	One-sample Wilcoxon test	`stats::wilcox.test()`
Robust	Bootstrap-*t* method for one-sample test	`WRS2::trimcibt()`
Bayesian	One-sample Student's *t*-test	`BayesFactor::ttestBF()`

Effect size estimation

Type	Effect size	CI available?	Function used
Parametric	Cohen's *d*, Hedge's *g*	Yes	`effectsize::cohens_d()`, `effectsize::hedges_g()`
Non-parametric	*r* (rank-biserial correlation)	Yes	`effectsize::rank_biserial()`
Robust	trimmed mean	Yes	`WRS2::trimcibt()`
Bayesian	difference	Yes	`bayestestR::describe_posterior()`

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

Examples

```
# for reproducibility
set.seed(123)

# ---------------------- parametric ----------------------

one_sample_test(mtcars, wt, test.value = 3)

# biased (Cohen's d) effect size
one_sample_test(mtcars, wt, test.value = 3, effsize.type = "d")

# ---------------------- non-parametric ------------------

one_sample_test(mtcars, wt, test.value = 3, type = "nonparametric")

# ---------------------- robust --------------------------

one_sample_test(mtcars, wt, test.value = 3, type = "robust")

# ---------------------- Bayesian ------------------------

one_sample_test(mtcars, wt, test.value = 3, type = "bayes")
```

---

pairwise_comparisons *Multiple pairwise comparison for one-way design*

Description

Calculate parametric, non-parametric, robust, and Bayes Factor pairwise comparisons between group levels with corrections for multiple testing.

Usage

```
pairwise_comparisons(
  data,
  x,
  y,
  subject.id = NULL,
  type = "parametric",
  paired = FALSE,
  var.equal = FALSE,
  tr = 0.2,
  bf.prior = 0.707,
  p.adjust.method = "holm",
  digits = 2L,
  exact = FALSE,
  ...
)
```

Arguments

data	A data frame (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix, table, array, etc.) will **not** be accepted. Additionally, grouped data frames from {dplyr} should be ungrouped before they are entered as data.
x	The grouping (or independent) variable from data. For repeated measures designs, see the note on ordering in subject.id.
y	The response (or outcome or dependent) variable from data.
subject.id	Relevant in case of a repeated measures or within-subjects design (i.e., paired = TRUE), it specifies the subject or repeated measures identifier. **Important**: If this argument is NULL (which is the default), observations are paired by their row order within each level of x (i.e., the data is assumed to be sorted in a subject-1, subject-2, ... pattern within every level). If the data is **not** sorted this way, the paired results will be silently incorrect, so it is safest to always specify subject.id.
type	A character specifying the type of statistical approach: • "parametric" • "nonparametric" • "robust" • "bayes" You can specify just the initial letter (e.g. "np" or "bf"). Matching is on the initial lowercase letter only, so any other value (including capitalized values such as "Bayes") falls back to "parametric" without a warning.
paired	Logical that decides whether the experimental design is repeated measures/within-subjects or between-subjects. The default is FALSE.
var.equal	a logical variable indicating whether to treat the two variances as being equal. If TRUE then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.
tr	Trim level for the mean when carrying out robust tests. In case of an error, try reducing the value of tr, which is by default set to 0.2. Lowering the value might help.
bf.prior	A number between 0.5 and 2 (default 0.707), the prior width to use in calculating Bayes factors and posterior estimates. In addition to numeric arguments, several named values are also recognized: "medium", "wide", and "ultrawide", corresponding to *r* scale values of 1/2, sqrt(2)/2, and 1, respectively. In case of an ANOVA, this value corresponds to scale for fixed effects.
p.adjust.method	Adjustment method for *p*-values for multiple comparisons. Possible methods are: "holm" (default), "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none".
digits	Number of digits for rounding or significant figures. May also be "signif" to return significant figures or "scientific" to return scientific notation. Control the number of digits by adding the value as suffix, e.g. digits = "scientific4" to have scientific notation with 4 decimal places, or digits = "signif5" for 5 significant figures (see also signif()).

exact	A logical indicating whether you want exact p-values to be computed. Relevant only when type = "nonparametric" (Default: FALSE).
...	Additional arguments passed to the underlying pairwise test function for the parametric and non-parametric tests (see the table below). Ignored for robust tests. For Bayesian tests, they are passed to the frequentist test (stats::pairwise.t.test() or PMCMRplus::gamesHowellTest()) that is run first to enumerate the pairs, so they do not change the Bayes factors, but unsupported arguments can still cause an error.

Value

The returned object is a tibble data frame with the additional class "statsExpressions". The exact set of columns depends on the test and, for functions that accept a type argument, on the chosen analysis (parametric, non-parametric, robust, or Bayesian). Any given call therefore returns *some* (not all) of the columns below.

Hypothesis testing

- statistic: the numeric value of a statistic
- df: the numeric value of a parameter being modeled (often degrees of freedom for the test)
- df.error and df: relevant only if the statistic in question has two degrees of freedom (e.g. anova)
- p.value: the *p*-value associated with the observed statistic (two-sided unless a one-sided alternative is requested, where supported)
- method: the name of the inferential statistical test

Effect size estimation

- effectsize: the name of the effect size
- estimate: estimated value of the effect size
- conf.level: the coverage level of the confidence/credible interval (e.g. 0.95); the interval itself spans conf.low to conf.high
- conf.low: lower bound for the effect size estimate
- conf.high: upper bound for the effect size estimate
- conf.method: method used to compute the confidence/credible interval
- conf.distribution: statistical distribution for the effect

Bayesian analysis (only when type = "bayes")

- bf10: Bayes factor for the alternative hypothesis relative to the null
- log_e_bf10: natural logarithm of the Bayes factor (present for most, but not all, Bayesian analyses)
- prior.distribution, prior.scale, prior.location: prior specification used to compute the Bayes factor and posterior estimates

Pairwise comparisons (for pairwise_comparisons() and pairwise_contingency_table())

- group1, group2: the two levels being compared

- `p.adjust.method`: the adjustment method used for multiple comparisons
- `p.value.adj`: the adjusted *p*-value; returned by `pairwise_contingency_table()`. Note that `pairwise_comparisons()` instead folds the adjusted value into `p.value` (and does not return a separate `p.value.adj` column)

Common columns

- `n.obs`: number of observations
- `expression`: a list-column of pre-formatted plotmath expressions; each element is a `language` object (not a character string) containing the statistical details, ready to be used in `{ggplot2}` (e.g. in `labs()` or `annotate()`)

For a per-function, column-by-column breakdown of the output (and an explanation of the internal `add_expression_col()` engine that builds the `expression` column), see the Return value schema article. For more examples, see the data frame output vignette.

Pairwise comparison tests

The table below provides summary about:

- statistical test carried out for inferential statistics
- type of effect size estimate and a measure of uncertainty for this estimate
- functions used internally to compute these details

between-subjects:

Hypothesis testing

Type	Equal variance?	Test	*p*-value adjustment?	Function used
Parametric	No	Games-Howell test	Yes	`PMCMRplus::gamesHowellTest()`
Parametric	Yes	Student's *t*-test	Yes	`stats::pairwise.t.test()`
Non-parametric	No	Dunn test	Yes	`PMCMRplus::kwAllPairsDunnTes`
Robust	No	Yuen's trimmed means test	Yes	`WRS2::lincon()`
Bayesian	NA	Student's *t*-test	NA	`BayesFactor::ttestBF()`

Effect size estimation

Not supported.

within-subjects:

Data requirement: Paired pairwise tests assume exactly **one observation per subject per condition**. If your data has multiple trials per cell, aggregate first (e.g., take the mean).

Hypothesis testing

Type	Test	*p*-value adjustment?	Function used
Parametric	Student's *t*-test	Yes	`stats::pairwise.t.test()`
Non-parametric	Durbin-Conover test	Yes	`PMCMRplus::durbinAllPairsTest()`
Robust	Yuen's trimmed means test	Yes	`WRS2::rmmcp()`
Bayesian	Student's *t*-test	NA	`BayesFactor::ttestBF()`

Effect size estimation

Not supported.

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

References

For more, see: https://www.indrapatil.com/ggstatsplot/articles/web_only/pairwise.html

Examples

```
# for reproducibility
set.seed(123)
library(statsExpressions)

#------------------- between-subjects design -----------------------------

# parametric
# if `var.equal = TRUE`, then Student's t-test will be run
pairwise_comparisons(
  data            = mtcars,
  x               = cyl,
  y               = wt,
  type            = "parametric",
  var.equal       = TRUE,
  paired          = FALSE,
  p.adjust.method = "none"
)

# if `var.equal = FALSE`, then Games-Howell test will be run
pairwise_comparisons(
  data            = mtcars,
  x               = cyl,
  y               = wt,
  type            = "parametric",
  var.equal       = FALSE,
  paired          = FALSE,
  p.adjust.method = "bonferroni"
)

# non-parametric (Dunn test)
pairwise_comparisons(
  data            = mtcars,
  x               = cyl,
  y               = wt,
  type            = "nonparametric",
```

```
  paired          = FALSE,
  p.adjust.method = "none"
)

# robust (Yuen's trimmed means *t*-test)
pairwise_comparisons(
  data            = mtcars,
  x               = cyl,
  y               = wt,
  type            = "robust",
  paired          = FALSE,
  p.adjust.method = "fdr"
)

# Bayes Factor (Student's *t*-test)
pairwise_comparisons(
  data   = mtcars,
  x      = cyl,
  y      = wt,
  type   = "bayes",
  paired = FALSE
)

#------------------- within-subjects design ----------------------------

# parametric (Student's *t*-test)
pairwise_comparisons(
  data            = bugs_long,
  x               = condition,
  y               = desire,
  subject.id      = subject,
  type            = "parametric",
  paired          = TRUE,
  p.adjust.method = "BH"
)

# non-parametric (Durbin-Conover test)
pairwise_comparisons(
  data            = bugs_long,
  x               = condition,
  y               = desire,
  subject.id      = subject,
  type            = "nonparametric",
  paired          = TRUE,
  p.adjust.method = "BY"
)

# robust (Yuen's trimmed means t-test)
pairwise_comparisons(
  data            = bugs_long,
  x               = condition,
  y               = desire,
  subject.id      = subject,
```

```
  type            = "robust",
  paired          = TRUE,
  p.adjust.method = "hommel"
)

# Bayes Factor (Student's *t*-test)
pairwise_comparisons(
  data       = bugs_long,
  x          = condition,
  y          = desire,
  subject.id = subject,
  type       = "bayes",
  paired     = TRUE
)
```

`pairwise_contingency_table` *Pairwise contingency table analyses*

Description

Pairwise Fisher's exact tests as post hoc tests for contingency table analyses, with effect sizes (Cramer's V) and p-value adjustment for multiple comparisons.

Usage

```
pairwise_contingency_table(
  data,
  x,
  y,
  counts = NULL,
  p.adjust.method = "holm",
  digits = 2L,
  conf.level = 0.95,
  alternative = "two.sided",
  ...
)
```

Arguments

`data`	A data frame (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix, table, array, etc.) will **not** be accepted. Additionally, grouped data frames from `{dplyr}` should be ungrouped before they are entered as `data`.
`x`	The variable to use as the **rows** in the contingency table.

y	The variable to use as the **columns** in the contingency table. Default is NULL. If NULL, one-sample proportion test (a goodness of fit test) will be run for the x variable.
counts	The variable in data containing counts, or NULL if each row represents a single observation.
p.adjust.method	Adjustment method for *p*-values for multiple comparisons. Possible methods are: "holm" (default), "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none".
digits	Number of digits for rounding or significant figures. May also be "signif" to return significant figures or "scientific" to return scientific notation. Control the number of digits by adding the value as suffix, e.g. digits = "scientific4" to have scientific notation with 4 decimal places, or digits = "signif5" for 5 significant figures (see also signif()).
conf.level	Scalar between 0 and 1 (default: 95% confidence/credible intervals, 0.95).
alternative	A character string specifying the alternative hypothesis; Controls the type of CI returned: "two.sided" (default, two-sided CI), "greater" or "less" (one-sided CI). Partial matching is allowed (e.g., "g", "l", "two"...). See section *One-Sided CIs* in the effectsize_CIs vignette.
...	Additional arguments passed to stats::fisher.test().

Value

The returned object is a tibble data frame with the additional class "statsExpressions". The exact set of columns depends on the test and, for functions that accept a type argument, on the chosen analysis (parametric, non-parametric, robust, or Bayesian). Any given call therefore returns *some* (not all) of the columns below.

Hypothesis testing

- statistic: the numeric value of a statistic
- df: the numeric value of a parameter being modeled (often degrees of freedom for the test)
- df.error and df: relevant only if the statistic in question has two degrees of freedom (e.g. anova)
- p.value: the *p*-value associated with the observed statistic (two-sided unless a one-sided alternative is requested, where supported)
- method: the name of the inferential statistical test

Effect size estimation

- effectsize: the name of the effect size
- estimate: estimated value of the effect size
- conf.level: the coverage level of the confidence/credible interval (e.g. 0.95); the interval itself spans conf.low to conf.high
- conf.low: lower bound for the effect size estimate
- conf.high: upper bound for the effect size estimate

- `conf.method`: method used to compute the confidence/credible interval
- `conf.distribution`: statistical distribution for the effect

Bayesian analysis (only when `type = "bayes"`)

- `bf10`: Bayes factor for the alternative hypothesis relative to the null
- `log_e_bf10`: natural logarithm of the Bayes factor (present for most, but not all, Bayesian analyses)
- `prior.distribution`, `prior.scale`, `prior.location`: prior specification used to compute the Bayes factor and posterior estimates

Pairwise comparisons (for `pairwise_comparisons()` and `pairwise_contingency_table()`)

- `group1`, `group2`: the two levels being compared
- `p.adjust.method`: the adjustment method used for multiple comparisons
- `p.value.adj`: the adjusted *p*-value; returned by `pairwise_contingency_table()`. Note that `pairwise_comparisons()` instead folds the adjusted value into `p.value` (and does not return a separate `p.value.adj` column)

Common columns

- `n.obs`: number of observations
- `expression`: a list-column of pre-formatted plotmath expressions; each element is a `language` object (not a character string) containing the statistical details, ready to be used in `{ggplot2}` (e.g. in `labs()` or `annotate()`)

For a per-function, column-by-column breakdown of the output (and an explanation of the internal `add_expression_col()` engine that builds the `expression` column), see the Return value schema article. For more examples, see the data frame output vignette.

Pairwise contingency table tests

The table below provides summary about:

- statistical test carried out for inferential statistics
- type of effect size estimate and a measure of uncertainty for this estimate
- functions used internally to compute these details

Hypothesis testing

Test	*p*-value adjustment?	Function used
Fisher's exact test	Yes	`stats::fisher.test()`

Effect size estimation

Effect size	CI available?	Function used
Cramer's *V*	Yes	`effectsize::cramers_v()`

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

Examples

```
# for reproducibility
set.seed(123)
library(statsExpressions)

# pairwise Fisher's exact tests with Holm adjustment
pairwise_contingency_table(
  data = mtcars,
  x = cyl,
  y = am,
  p.adjust.method = "holm"
)

# with counts data and Bonferroni adjustment
pairwise_contingency_table(
  data = as.data.frame(Titanic),
  x = Class,
  y = Survived,
  counts = Freq,
  p.adjust.method = "bonferroni"
)

# no p-value adjustment
pairwise_contingency_table(
  data = mtcars,
  x = cyl,
  y = am,
  p.adjust.method = "none"
)
```

---

tidy_model_expressions

Expressions with statistics for tidy regression data frames

Description

Expressions with statistics for tidy regression data frames

Usage

```
tidy_model_expressions(
  data,
  statistic = NULL,
```

```
  digits = 2L,
  effsize.type = "omega",
  ...
)
```

Arguments

`data`	A tidy data frame from regression model object (see `tidy_model_parameters()`). It must contain a `term` column identifying each row.
`statistic`	Which statistic is to be displayed (either `"t"`, `"f"`, `"z"`, or `"chi"`) in the expression. This argument must be specified.
`digits`	Number of digits for rounding or significant figures. May also be `"signif"` to return significant figures or `"scientific"` to return scientific notation. Control the number of digits by adding the value as suffix, e.g. `digits = "scientific4"` to have scientific notation with 4 decimal places, or `digits = "signif5"` for 5 significant figures (see also `signif()`).
`effsize.type`	Type of effect size needed for *parametric* tests. The argument can be `"eta"` (partial eta-squared) or `"omega"` (partial omega-squared).
`...`	Currently ignored.

Details

When any of the necessary numeric column values (`estimate`, `statistic`, `p.value`) are missing, for these rows, a `NULL` is returned instead of an expression with empty strings.

Value

The input data frame (as a tibble) with an additional `expression` list-column.

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

Examples

```
# setup
set.seed(123)
library(statsExpressions)

# extract a tidy data frame
df <- tidy_model_parameters(lm(wt ~ am * cyl, mtcars))

# create a column containing expression; the expression will depend on `statistic`
tidy_model_expressions(df, statistic = "t")
tidy_model_expressions(df, statistic = "z")
tidy_model_expressions(df, statistic = "chi")

# f-statistic (requires a data frame with `df`, `df.error`, and effect-size columns)
df_f <- tidy_model_parameters(
```

```
  aov(wt ~ cyl, mtcars),
  es_type = "omega",
  table_wide = TRUE
)
tidy_model_expressions(df_f, statistic = "f")
df_f_eta <- tidy_model_parameters(
  aov(wt ~ cyl, mtcars),
  es_type = "eta",
  table_wide = TRUE
)
tidy_model_expressions(df_f_eta, statistic = "f", effsize.type = "eta")
```

`tidy_model_parameters` *Convert* {parameters} *package output to* {tidyverse} *conventions*

Description

Runs `parameters::model_parameters()` on a model object and standardizes the result to {broom}-style column names (e.g. `estimate`, `std.error`, `conf.low`, `conf.high`, `statistic`, `df.error`, `p.value`). Bayes factors are returned in a `bf10` column, along with their natural logarithm in `log_e_bf10`. For Bayesian ANOVA designs ({BayesFactor} linear models), the estimate is replaced by the Bayesian *R*-squared.

Usage

```
tidy_model_parameters(model, ...)
```

Arguments

`model`	Statistical Model.
`...`	Arguments passed to or from other methods. Non-documented arguments are • `digits`, `p_digits`, `ci_digits` and `footer_digits` to set the number of digits for the output. `groups` can be used to group coefficients. These arguments will be passed to the print-method, or can directly be used in `print()`, see documentation in `print.parameters_model()`. • If `s_value = TRUE`, the p-value will be replaced by the S-value in the output (cf. *Rafi and Greenland 2020*). • pd adds an additional column with the *probability of direction* (see `bayestestR::p_direction()` for details). Furthermore, see 'Examples' in `model_parameters.default()`. • For developers, whose interest mainly is to get a "tidy" data frame of model summaries, it is recommended to set `pretty_names = FALSE` to speed up computation of the summary table.

Value

A tibble with one row per parameter.

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

Examples

```
model <- lm(mpg ~ wt + cyl, data = mtcars)
tidy_model_parameters(model)
```

---

two_sample_test *Two-sample tests*

Description

Parametric, non-parametric, robust, and Bayesian two-sample tests.

Usage

```
two_sample_test(
  data,
  x,
  y,
  subject.id = NULL,
  type = "parametric",
  paired = FALSE,
  alternative = "two.sided",
  digits = 2L,
  conf.level = 0.95,
  effsize.type = "g",
  var.equal = FALSE,
  bf.prior = 0.707,
  tr = 0.2,
  nboot = 100L,
  exact = FALSE,
  ...
)
```

Arguments

data	A data frame (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix, table, array, etc.) will **not** be accepted. Additionally, grouped data frames from {dplyr} should be ungrouped before they are entered as data.
x	The grouping (or independent) variable from data. For repeated measures designs, see the note on ordering in subject.id.
y	The response (or outcome or dependent) variable from data.

`subject.id` Relevant in case of a repeated measures or within-subjects design (i.e., `paired = TRUE`), it specifies the subject or repeated measures identifier. **Important**: If this argument is `NULL` (which is the default), observations are paired by their row order within each level of `x` (i.e., the data is assumed to be sorted in a subject-1, subject-2, ... pattern within every level). If the data is **not** sorted this way, the paired results will be silently incorrect, so it is safest to always specify `subject.id`.

`type` A character specifying the type of statistical approach:

- `"parametric"`
- `"nonparametric"`
- `"robust"`
- `"bayes"`

You can specify just the initial letter (e.g. `"np"` or `"bf"`). Matching is on the initial lowercase letter only, so any other value (including capitalized values such as `"Bayes"`) falls back to `"parametric"` without a warning.

`paired` Logical that decides whether the experimental design is repeated measures/within-subjects or between-subjects. The default is `FALSE`.

`alternative` a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.

`digits` Number of digits for rounding or significant figures. May also be `"signif"` to return significant figures or `"scientific"` to return scientific notation. Control the number of digits by adding the value as suffix, e.g. `digits = "scientific4"` to have scientific notation with 4 decimal places, or `digits = "signif5"` for 5 significant figures (see also `signif()`).

`conf.level` Scalar between `0` and `1` (default: `95%` confidence/credible intervals, `0.95`).

`effsize.type` Type of effect size needed for *parametric* tests. The argument can be `"d"` (for Cohen's *d*) or `"g"` (for Hedge's *g*).

`var.equal` a logical variable indicating whether to treat the two variances as being equal. If `TRUE` then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.

`bf.prior` A number between `0.5` and `2` (default `0.707`), the prior width to use in calculating Bayes factors and posterior estimates. In addition to numeric arguments, several named values are also recognized: `"medium"`, `"wide"`, and `"ultrawide"`, corresponding to *r* scale values of `1/2`, `sqrt(2)/2`, and `1`, respectively. In case of an ANOVA, this value corresponds to scale for fixed effects.

`tr` Trim level for the mean when carrying out `robust` tests. In case of an error, try reducing the value of `tr`, which is by default set to `0.2`. Lowering the value might help.

`nboot` Number of bootstrap samples for computing confidence interval for the effect size (Default: `100L`).

`exact` A logical indicating whether you want exact p-values to be computed. Relevant only when `type = "nonparametric"` (Default: `FALSE`).

`...` Currently ignored.

Value

The returned object is a tibble data frame with the additional class `"statsExpressions"`. The exact set of columns depends on the test and, for functions that accept a `type` argument, on the chosen analysis (parametric, non-parametric, robust, or Bayesian). Any given call therefore returns *some* (not all) of the columns below.

Hypothesis testing

- `statistic`: the numeric value of a statistic
- `df`: the numeric value of a parameter being modeled (often degrees of freedom for the test)
- `df.error` and `df`: relevant only if the statistic in question has two degrees of freedom (e.g. anova)
- `p.value`: the *p*-value associated with the observed statistic (two-sided unless a one-sided `alternative` is requested, where supported)
- `method`: the name of the inferential statistical test

Effect size estimation

- `effectsize`: the name of the effect size
- `estimate`: estimated value of the effect size
- `conf.level`: the coverage level of the confidence/credible interval (e.g. `0.95`); the interval itself spans `conf.low` to `conf.high`
- `conf.low`: lower bound for the effect size estimate
- `conf.high`: upper bound for the effect size estimate
- `conf.method`: method used to compute the confidence/credible interval
- `conf.distribution`: statistical distribution for the effect

Bayesian analysis (only when `type = "bayes"`)

- `bf10`: Bayes factor for the alternative hypothesis relative to the null
- `log_e_bf10`: natural logarithm of the Bayes factor (present for most, but not all, Bayesian analyses)
- `prior.distribution`, `prior.scale`, `prior.location`: prior specification used to compute the Bayes factor and posterior estimates

Pairwise comparisons (for `pairwise_comparisons()` and `pairwise_contingency_table()`)

- `group1`, `group2`: the two levels being compared
- `p.adjust.method`: the adjustment method used for multiple comparisons
- `p.value.adj`: the adjusted *p*-value; returned by `pairwise_contingency_table()`. Note that `pairwise_comparisons()` instead folds the adjusted value into `p.value` (and does not return a separate `p.value.adj` column)

Common columns

- `n.obs`: number of observations

- `expression`: a list-column of pre-formatted plotmath expressions; each element is a `language` object (not a character string) containing the statistical details, ready to be used in `{ggplot2}` (e.g. in `labs()` or `annotate()`)

For a per-function, column-by-column breakdown of the output (and an explanation of the internal `add_expression_col()` engine that builds the `expression` column), see the Return value schema article. For more examples, see the data frame output vignette.

Two-sample tests

The table below provides summary about:

- statistical test carried out for inferential statistics
- type of effect size estimate and a measure of uncertainty for this estimate
- functions used internally to compute these details

between-subjects:

Hypothesis testing

Type	No. of groups	Test	Function used
Parametric	2	Student's or Welch's *t*-test	`stats::t.test()`
Non-parametric	2	Mann-Whitney *U* test	`stats::wilcox.test()`
Robust	2	Yuen's test for trimmed means	`WRS2::yuen()`
Bayesian	2	Student's *t*-test	`BayesFactor::ttestBF()`

Effect size estimation

Type	No. of groups	Effect size	CI available?	Function used
Parametric	2	Cohen's *d*, Hedge's *g*	Yes	`effectsize:`
Non-parametric	2	*r* (rank-biserial correlation)	Yes	`effectsize:`
Robust	2	Algina-Keselman-Penfield robust standardized difference	Yes	`WRS2::akp.e`
Bayesian	2	difference	Yes	`bayestestR:`

within-subjects:

Data requirement: Paired tests assume exactly **one observation per subject per condition**. If your data has multiple trials per cell, aggregate first (e.g., take the mean).

Hypothesis testing

Type	No. of groups	Test	Function used
Parametric	2	Student's *t*-test	`stats::t.test()`
Non-parametric	2	Wilcoxon signed-rank test	`stats::wilcox.test()`
Robust	2	Yuen's test on trimmed means for dependent samples	`WRS2::yuend()`
Bayesian	2	Student's *t*-test	`BayesFactor::ttestBF()`

Effect size estimation

Type	No. of groups	Effect size	CI available?	Function used
Parametric	2	Cohen's *d*, Hedge's *g*	Yes	`effectsize:`
Non-parametric	2	*r* (rank-biserial correlation)	Yes	`effectsize:`
Robust	2	Algina-Keselman-Penfield robust standardized difference	Yes	`WRS2::wmcpA`
Bayesian	2	difference	Yes	`bayestestR:`

Citation

Patil, I., (2021). statsExpressions: R Package for Tidy Dataframes and Expressions with Statistical Details. Journal of Open Source Software, 6(61), 3236, https://doi.org/10.21105/joss.03236

Examples

```
# ----------------------- within-subjects -------------------------------------

# data
df <- dplyr::filter(bugs_long, condition %in% c("LDLF", "LDHF"))

# for reproducibility
set.seed(123)

# ----------------------- parametric ------------------------------------------

two_sample_test(
  df,
  condition,
  desire,
  subject.id = subject,
  paired = TRUE,
  type = "parametric"
)

# ----------------------- non-parametric --------------------------------------

two_sample_test(
  df,
  condition,
  desire,
  subject.id = subject,
  paired = TRUE,
  type = "nonparametric"
)

# ----------------------- robust ----------------------------------------------

two_sample_test(
  df,
  condition,
  desire,
  subject.id = subject,
  paired = TRUE,
```

```
  type = "robust"
)

# ---------------------- Bayesian --------------------------------------

two_sample_test(
  df,
  condition,
  desire,
  subject.id = subject,
  paired = TRUE,
  type = "bayes"
)

# ---------------------- between-subjects ------------------------------------

# for reproducibility
set.seed(123)

# ---------------------- parametric ------------------------------------

# unequal variance
two_sample_test(ToothGrowth, supp, len, type = "parametric")

# equal variance
two_sample_test(ToothGrowth, supp, len, type = "parametric", var.equal = TRUE)

# biased (Cohen's d) effect size
two_sample_test(ToothGrowth, supp, len, type = "parametric", effsize.type = "d")

# ---------------------- non-parametric ----------------------------------

two_sample_test(ToothGrowth, supp, len, type = "nonparametric")

# ---------------------- robust ------------------------------------------

two_sample_test(ToothGrowth, supp, len, type = "robust")

# ---------------------- Bayesian --------------------------------------

two_sample_test(ToothGrowth, supp, len, type = "bayes")
```

Index

∗ **datasets**
    bugs_long, 4
    iris_long, 18
    movies_long, 24

add_expression_col, 2

BayesFactor::contingencyTableBF(), *10*
bayestestR::p_direction(), *45*
bugs_long, 4

centrality_description, 5
contingency_table, 8
corr_test, 14

datawizard::describe_distribution(), *6*

extract_stats_type, 17

iris_long, 18

long_to_wide_converter, 19

meta_analysis, 20
metaBMA::meta_random(), *21*
metafor::rma(), *21*
metaplus::metaplus(), *21*
model_parameters.default(), *45*
movies_long, 24

one_sample_test, 30
oneway_anova, 25

pairwise_comparisons, 34
pairwise_contingency_table, 40
parameters::model_parameters(), *45*
print.parameters_model(), *45*

signif(), *6*, *9*, *14*, *21*, *26*, *31*, *35*, *41*, *44*, *47*
stats::fisher.test(), *41*
stats_type_switch *(*extract_stats_type*)*, 17

tidy_model_expressions, 43
tidy_model_parameters, 45
tidy_model_parameters(), *44*
two_sample_test, 46