Package 'statsExpressions'

January 14, 2023

Type Package

Title Tidy Dataframes and Expressions with Statistical Details

Version 1.4.0

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 GPL-3 | file LICENSE

URL https://indrajeetpatil.github.io/statsExpressions/, https://github.com/IndrajeetPatil/statsExpressions

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

Depends R (>= 4.0.0)

Imports BayesFactor (>= 0.9.12-4.4), correlation (>= 0.8.3), datawizard (>= 0.6.5), dplyr, effectsize (>= 0.8.2), glue, insight (>= 0.18.8), magrittr, parameters (>= 0.20.1), performance (>= 0.10.2), purrr (>= 1.0.1), rlang, stats, tibble, tidyr, withr, WRS2 (>= 1.1-4), zeallot

Suggests afex (>= 1.2-0), ggplot2, knitr, metaBMA, metafor, metaplus, PMCMRplus, rmarkdown, survival, testthat (>= 3.1.6), utils

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.3.9000

Config/testthat/edition 3

NeedsCompilation no

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

Repository CRAN

Date/Publication 2023-01-14 20:20:02 UTC

R topics documented:

`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` function 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"]))),
  k = 2L,
  k.df = 0L,
  k.df.error = k.df,
  ...
)
```

Arguments

`data`	A data frame containing details from the statistical analysis and should contain some or all of the 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); note that if there are no degrees of freedom (e.g., for non-parametric tests), this column will be irrelevant. • *df*: relevant only if the statistic in question has two degrees of freedom. • *p.value*: the two-sided *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 `bayesian = TRUE`).
`paired`	Logical that decides whether the experimental design is repeated measures/within-subjects or between-subjects. The default is `FALSE`.
`statistic.text`	A character that specifies the relevant test statistic. For example, for tests with *t*-statistic, `statistic.text = "t"`.
`effsize.text`	A character that specifies the relevant effect size.
`prior.type`	The type of prior.
`n`	An integer specifying the sample size used for the test.
`n.text`	A character 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`. If you wish to customize this further, you will need to provide object of `language` type.

k Number of digits after decimal point (should be an integer) (Default: k = 2L).

k.df, k.df.error
Number of decimal places to display for the parameters (default: 0L).

... Currently ignored.

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"])),
  k              = 3L,
  k.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 freighteningness and low disgustingness; **LFHD**: low freighteningness and high disgustingness; **HFHD**: high freighteningness and low disgustingness; **HFHD**: high freighteningness and high disgustingness).
- 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 freighteningness (low, high) and disgustingness (low, high). Each participant rates their attitudes towards all anthropods. Subset of the data reported by Ryan et al. (2013).

Source

https://www.sciencedirect.com/science/article/pii/S0747563213000277

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 = NULL,
  tr = 0.2,
  k = 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.
`conf.level`	Scalar between `0` and `1` (default: `95%` confidence/credible intervals, `0.95`). If `NULL`, no confidence intervals will be computed.
`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.
`k`	Number of digits after decimal point (should be an integer) (Default: `k = 2L`).
`...`	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.

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()`

Examples

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

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

centrality_description(iris, Species, Sepal.Length)

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

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

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

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

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

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

---

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,
  k = 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. Otherwise association test will be carried out.
paired	Logical indicating whether data came from a within-subjects or repeated measures design study (Default: FALSE). If TRUE, McNemar's test expression will be returned. If FALSE, Pearson's chi-square test will be returned.
type	A character specifying the type of statistical approach: • "parametric" • "nonparametric" • "robust" • "bayes" You can specify just the initial letter.
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. This means if there are two levels this will be ratio = c(0.5,0.5) or if there are four levels this will be ratio = c(0.25,0.25,0.25,0.25), etc.
k	Number of digits after decimal point (should be an integer) (Default: k = 2L).
conf.level	Scalar between 0 and 1 (default: 95% confidence/credible intervals, 0.95). If NULL, no confidence intervals will be computed.
sampling.plan	Character describing the sampling plan. Possible options are "indepMulti" (independent multinomial; default), "poisson", "jointMulti" (joint multinomial), "hypergeom" (hypergeometric). For more, see ?BayesFactor::contingencyTableBF().
fixed.margin	For the independent multinomial sampling plan, which margin is fixed ("rows" or "cols"). Defaults to "rows".
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.
...	Additional arguments (currently ignored).

Value

The returned tibble data frame can contain some or all of the following columns (the exact columns will depend on the statistical test):

- `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 two-sided *p*-value associated with the observed statistic
- `method`: the name of the inferential statistical test
- `estimate`: estimated value of the effect size
- `conf.low`: lower bound for the effect size estimate
- `conf.high`: upper bound for the effect size estimate
- `conf.level`: width of the confidence interval
- `conf.method`: method used to compute confidence interval
- `conf.distribution`: statistical distribution for the effect
- `effectsize`: the name of the effect size
- `n.obs`: number of observations
- `expression`: pre-formatted expression containing statistical details

For examples, see 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

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

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

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

# ------------------------ Frequentist -----------------------------

# association test
contingency_table(
  data   = mtcars,
  x      = am,
  y      = vs,
  paired = FALSE
)

# goodness-of-fit test
contingency_table(
  data   = as.data.frame(HairEyeColor),
  x      = Eye,
  counts = Freq,
  ratio  = c(0.2, 0.2, 0.3, 0.3)
)

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

# association test
contingency_table(
  data   = mtcars,
  x      = am,
  y      = vs,
  paired = FALSE,
  type   = "bayes"
)
```

```
# goodness-of-fit test
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",
  k = 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.
k	Number of digits after decimal point (should be an integer) (Default: k = 2L).
conf.level	Scalar between 0 and 1 (default: 95% confidence/credible intervals, 0.95). If NULL, no confidence intervals will be computed.
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 tibble data frame can contain some or all of the following columns (the exact columns will depend on the statistical test):

- 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 two-sided *p*-value associated with the observed statistic
- method: the name of the inferential statistical test
- estimate: estimated value of the effect size
- conf.low: lower bound for the effect size estimate
- conf.high: upper bound for the effect size estimate
- conf.level: width of the confidence interval
- conf.method: method used to compute confidence interval
- conf.distribution: statistical distribution for the effect
- effectsize: the name of the effect size
- n.obs: number of observations
- expression: pre-formatted expression containing statistical details

For examples, see 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()`

Examples

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

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

corr_test(mtcars, wt, mpg)

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

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

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

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

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

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

`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 5 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 `"Pepal"`).
- 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 with `NA`*s removed*

Description

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

It does not make sense to spread the data frame to wide format when the measure is not repeated, so if `paired = TRUE`, `spread` argument will be ignored.

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`. In case of a repeated measures or within-subjects design, if `subject.id` argument is not available or not explicitly specified, the function assumes that the data has already been sorted by such an id by the user and creates an internal identifier. So if your data is **not** sorted, the results *can* be inaccurate when there are more than two levels

	in x and there are NAs present. The data is expected to be sorted by user in subject-1,subject-2, ..., pattern.
y	The response (or outcome or dependent) variable from data.
subject.id	Relevant in case of a repeated measures or within-subjects design (paired = TRUE, i.e.), it specifies the subject or repeated measures identifier. **Important**: Note that if this argument is NULL (which is the default), the function assumes that the data has already been sorted by such an id by the user and creates an internal identifier. So if your data is **not** sorted and you leave this argument unspecified, the results *can* be inaccurate when there are more than two levels in x and there are NAs present.
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). Relevant only if paired = TRUE.
...	Currently ignored.

Value

A data frame with NAs removed while respecting the between-or-within-subjects nature of the dataset.

Examples

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

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

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

`meta_analysis` *Random-effects meta-analysis*

Description

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

Usage

```
meta_analysis(
  data,
  type = "parametric",
  random = "mixture",
  k = 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) or `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.
`random`	The type of random effects distribution. One of "normal", "t-dist", "mixture", for standard normal, t-distribution or mixture of normals respectively.
`k`	Number of digits after decimal point (should be an integer) (Default: `k = 2L`).
`conf.level`	Scalar between `0` and `1` (default: `95%` confidence/credible intervals, `0.95`). If `NULL`, no confidence intervals will be computed.
`...`	Additional arguments passed to the respective meta-analysis function.

Value

The returned tibble data frame can contain some or all of the following columns (the exact columns will depend on the statistical test):

- `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 two-sided *p*-value associated with the observed statistic
- `method`: the name of the inferential statistical test
- `estimate`: estimated value of the effect size
- `conf.low`: lower bound for the effect size estimate
- `conf.high`: upper bound for the effect size estimate
- `conf.level`: width of the confidence interval
- `conf.method`: method used to compute confidence interval
- `conf.distribution`: statistical distribution for the effect
- `effectsize`: the name of the effect size
- `n.obs`: number of observations
- `expression`: pre-formatted expression containing statistical details

For examples, see 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	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()`

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

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

```
# a data frame with estimates and standard errors
# (`mag` dataset from `{metaplus}`)
df <- tibble::tribble(
  ~study, ~estimate, ~std.error,
  "Abraham", -0.83, 1.247,
  "Bertschat", -1.056, 0.414,
  "Ceremuzynski", -1.278, 0.808,
  "Feldstedt", -0.043, 1.429,
  "Golf", 0.223, 0.489,
  "ISIS-4", -2.407, 1.072,
  "LIMIT-2", -1.28, 1.193,
  "Morton", -1.191, 1.661,
  "Pereira", -0.695, 0.536,
  "Rasmussen", -2.208, 1.109,
  "Schechter", -2.038, 0.78,
  "Schechter 1", -0.85, 0.618,
  "Schechter 2", -0.793, 0.625,
  "Singh", -0.299, 0.146,
  "Smith", -1.57, 0.574,
  "Thogersen", 0.057, 0.031
)

# parametric
meta_analysis(df)


# robust
meta_analysis(df, type = "random", random = "normal")


# Bayesian
meta_analysis(df, type = "bayes")
```

`movies_long` *Movie information and user ratings from IMDB.com (long format).*

Description

Movie information and user ratings from IMDB.com (long format).

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. Different genres of movies (action, animation, comedy, drama, documentary, romance, short).

Details

Modified dataset from `ggplot2movies` package.

The internet movie database, https://imdb.com/, is a website devoted to collecting movie data supplied by studios and fans. It claims to be the biggest movie database on the web and is run by amazon.

Movies were are identical to those selected for inclusion in movies_wide but this dataset has been constructed such that every movie appears in one and only one genre category.

Source

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

Examples

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

`movies_wide` *Movie information and user ratings from IMDB.com (wide format).*

Description

Movie information and user ratings from IMDB.com (wide format).

Usage

```
movies_wide
```

Format

A data frame with 1,579 rows and 13 variables

- title. Title of the movie.
- year. Year of release.
- budget. Total budget in millions of US dollars
- length. Length in minutes.
- rating. Average IMDB user rating.
- votes. Number of IMDB users who rated this movie.
- mpaa. MPAA rating.
- action, animation, comedy, drama, documentary, romance, short. Binary variables representing if movie was classified as belonging to that genre.
- NumGenre. The number of different genres a film was classified in an integer between one and four.

Details

Modified dataset from `ggplot2movies` package.

The internet movie database, https://imdb.com/, is a website devoted to collecting movie data supplied by studios and fans. It claims to be the biggest movie database on the web and is run by amazon.

Movies were selected for inclusion if they had a known length and had been rated by at least one IMDB user. Small categories such as documentaries and NC-17 movies were removed.

Source

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

Examples

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

---

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,
  k = 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`. In case of a repeated measures or within-subjects design, if `subject.id` argument is not available or not explicitly specified, the function assumes that the data has already been sorted by such an id by the user and creates an internal identifier. So if your data is **not** sorted, the results *can* be inaccurate when there are more than two levels in `x` and there are `NA`s present. The data is expected to be sorted by user in subject-1,subject-2, ..., pattern.
`y`	The response (or outcome or dependent) variable from `data`.
`subject.id`	Relevant in case of a repeated measures or within-subjects design (`paired = TRUE`, i.e.), it specifies the subject or repeated measures identifier. **Important**: Note that if this argument is `NULL` (which is the default), the function assumes that the data has already been sorted by such an id by the user and creates an internal identifier. So if your data is **not** sorted and you leave this argument unspecified, the results *can* be inaccurate when there are more than two levels in `x` and there are `NA`s present.
`type`	A character specifying the type of statistical approach: • `"parametric"` • `"nonparametric"` • `"robust"` • `"bayes"` You can specify just the initial letter.

paired	Logical that decides whether the experimental design is repeated measures/within-subjects or between-subjects. The default is FALSE.
k	Number of digits after decimal point (should be an integer) (Default: k = 2L).
conf.level	Scalar between 0 and 1 (default: 95% confidence/credible intervals, 0.95). If NULL, no confidence intervals will be computed.
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 tibble data frame can contain some or all of the following columns (the exact columns will depend on the statistical test):

- 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 two-sided *p*-value associated with the observed statistic
- method: the name of the inferential statistical test
- estimate: estimated value of the effect size
- conf.low: lower bound for the effect size estimate
- conf.high: upper bound for the effect size estimate
- conf.level: width of the confidence interval
- conf.method: method used to compute confidence interval
- conf.distribution: statistical distribution for the effect
- effectsize: the name of the effect size
- n.obs: number of observations
- expression: pre-formatted expression containing statistical details

For examples, see 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()`
Bayes Factor	> 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()`
Bayes Factor	> 2	Bayesian R-squared	Yes	`performance::r2_bayes()`

within-subjects:

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::friedm`
Robust	> 2	Heteroscedastic one-way repeated measures ANOVA for trimmed means	`WRS2::rmanov`
Bayes Factor	> 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:`
Bayes Factor	> 2	Bayesian R-squared	Yes	`perfo`

Examples

```
# for reproducibility
set.seed(123)
library(statsExpressions)
suppressPackageStartupMessages(library(afex)) # for within-subjects parametric ANOVA

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

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

# 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",
  k = 2L,
```

```
  conf.level = 0.95,
  tr = 0.2,
  bf.prior = 0.707,
  effsize.type = "g",
  ...
)
```

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.
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.
k	Number of digits after decimal point (should be an integer) (Default: k = 2L).
conf.level	Scalar between 0 and 1 (default: 95% confidence/credible intervals, 0.95). If NULL, no confidence intervals will be computed.
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*).
...	Currently ignored.

Value

The returned tibble data frame can contain some or all of the following columns (the exact columns will depend on the statistical test):

- 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 two-sided *p*-value associated with the observed statistic
- `method`: the name of the inferential statistical test
- `estimate`: estimated value of the effect size
- `conf.low`: lower bound for the effect size estimate
- `conf.high`: upper bound for the effect size estimate
- `conf.level`: width of the confidence interval
- `conf.method`: method used to compute confidence interval
- `conf.distribution`: statistical distribution for the effect
- `effectsize`: the name of the effect size
- `n.obs`: number of observations
- `expression`: pre-formatted expression containing statistical details

For examples, see 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()`
Bayes Factor	difference	Yes	`bayestestR::describe_posterior()`

Examples

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

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

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

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

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

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

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

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

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

`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",
  k = 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 from data. In case of a repeated measures or within-subjects design, if subject.id argument is not available or not explicitly specified, the function assumes that the data has already been sorted by such an id by the user and creates an internal identifier. So if your data is **not** sorted, the results *can* be inaccurate when there are more than two levels in x and there are NAs present. The data is expected to be sorted by user in subject-1,subject-2, ..., pattern.
y	The response (or outcome or dependent) variable from data.
subject.id	Relevant in case of a repeated measures or within-subjects design (paired = TRUE, i.e.), it specifies the subject or repeated measures identifier. **Important**: Note that if this argument is NULL (which is the default), the function assumes that the data has already been sorted by such an id by the user and creates an internal identifier. So if your data is **not** sorted and you leave this argument unspecified, the results *can* be inaccurate when there are more than two levels in x and there are NAs present.
type	A character specifying the type of statistical approach: • "parametric" • "nonparametric" • "robust" • "bayes" You can specify just the initial letter.
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".
k	Number of digits after decimal point (should be an integer) (Default: k = 2L).
...	Additional arguments passed to other methods.

Value

The returned tibble data frame can contain some or all of the following columns (the exact columns will depend on the statistical test):

- `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 two-sided *p*-value associated with the observed statistic
- `method`: the name of the inferential statistical test
- `estimate`: estimated value of the effect size
- `conf.low`: lower bound for the effect size estimate
- `conf.high`: upper bound for the effect size estimate
- `conf.level`: width of the confidence interval
- `conf.method`: method used to compute confidence interval
- `conf.distribution`: statistical distribution for the effect
- `effectsize`: the name of the effect size
- `n.obs`: number of observations
- `expression`: pre-formatted expression containing statistical details

For examples, see 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:
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.

References

For more, see: https://indrajeetpatil.github.io/ggstatsplot/articles/web_only/pairwise.html

Examples

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

#------------------- 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
)
```

`p_adjust_text` p-*value adjustment method text*

Description

Preparing text to describe which *p*-value adjustment method was used

Usage

```
p_adjust_text(p.adjust.method)
```

Arguments

`p.adjust.method`
: Adjustment method for *p*-values for multiple comparisons. Possible methods are: `"holm"` (default), `"hochberg"`, `"hommel"`, `"bonferroni"`, `"BH"`, `"BY"`, `"fdr"`, `"none"`.

Value

Standardized text description for what method was used.

Examples

```
p_adjust_text("none")
p_adjust_text("BY")
```

stats_type_switch *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.

Usage

```
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.

Examples

```
stats_type_switch("p")
stats_type_switch("bf")
```

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

Arguments

data	A tidy data frame from regression model object (see statsExpressions::tidy_model_parameters()).
statistic	Which statistic is to be displayed (either "t" or "f"or "z" or "chi") in the expression.
k	Number of digits after decimal point (should be an integer) (Default: k = 2L).
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.

Note

This is an **experimental** function and may change in the future. Please do not use it yet in your workflow.

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")
```

---

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

Description

Convert {parameters} package output to {tidyverse} conventions

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. group can also be passed to the print() method. See details in print.parameters_model() and 'Examples' in model_parameters.default().

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",
  k = 2L,
  conf.level = 0.95,
  effsize.type = "g",
  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. In case of a repeated measures or within-subjects design, if subject.id argument is not available or not explicitly specified, the function assumes that the data has already been sorted by such an id by the user and creates an internal identifier. So if your data is **not** sorted, the results *can* be inaccurate when there are more than two levels in x and there are NAs present. The data is expected to be sorted by user in subject-1,subject-2, ..., pattern.
y	The response (or outcome or dependent) variable from data.
subject.id	Relevant in case of a repeated measures or within-subjects design (paired = TRUE, i.e.), it specifies the subject or repeated measures identifier. **Important**: Note that if this argument is NULL (which is the default), the function assumes that the data has already been sorted by such an id by the user and creates an internal identifier. So if your data is **not** sorted and you leave this argument unspecified, the results *can* be inaccurate when there are more than two levels in x and there are NAs present.
type	A character specifying the type of statistical approach: • "parametric" • "nonparametric" • "robust" • "bayes" You can specify just the initial letter.
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.
k	Number of digits after decimal point (should be an integer) (Default: k = 2L).
conf.level	Scalar between 0 and 1 (default: 95% confidence/credible intervals, 0.95). If NULL, no confidence intervals will be computed.
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).
...	Currently ignored.

Value

The returned tibble data frame can contain some or all of the following columns (the exact columns will depend on the statistical test):

- `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 two-sided *p*-value associated with the observed statistic
- `method`: the name of the inferential statistical test
- `estimate`: estimated value of the effect size
- `conf.low`: lower bound for the effect size estimate
- `conf.high`: upper bound for the effect size estimate
- `conf.level`: width of the confidence interval
- `conf.method`: method used to compute confidence interval
- `conf.distribution`: statistical distribution for the effect
- `effectsize`: the name of the effect size
- `n.obs`: number of observations
- `expression`: pre-formatted expression containing statistical details

For examples, see 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:

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:`

Examples

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

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

# between-subjects design
two_sample_test(
  data = sleep,
  x    = group,
  y    = extra,
  type = "p"
)

# within-subjects design
two_sample_test(
  data       = dplyr::filter(bugs_long, condition %in% c("HDHF", "HDLF")),
  x          = condition,
  y          = desire,
  paired     = TRUE,
  subject.id = subject,
  type       = "p"
)
```

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

# between-subjects design
two_sample_test(
  data = sleep,
  x    = group,
  y    = extra,
  type = "np"
)

# within-subjects design
two_sample_test(
  data       = dplyr::filter(bugs_long, condition %in% c("HDHF", "HDLF")),
  x          = condition,
  y          = desire,
  paired     = TRUE,
  subject.id = subject,
  type       = "np"
)

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

# between-subjects design
two_sample_test(
  data = sleep,
  x    = group,
  y    = extra,
  type = "r"
)

# within-subjects design
two_sample_test(
  data       = dplyr::filter(bugs_long, condition %in% c("HDHF", "HDLF")),
  x          = condition,
  y          = desire,
  paired     = TRUE,
  subject.id = subject,
  type       = "r"
)

#' # Bayesian ------------------------------

# between-subjects design
two_sample_test(
  data = sleep,
  x    = group,
  y    = extra,
  type = "bayes"
)

# within-subjects design
two_sample_test(
  data       = dplyr::filter(bugs_long, condition %in% c("HDHF", "HDLF")),
```

```
  x          = condition,
  y          = desire,
  paired     = TRUE,
  subject.id = subject,
  type       = "bayes"
)
```

Index