Package 'statsExpressions'

December 1, 2020

Type Package

Title Expressions and Dataframes with Statistical Details

Version 0.6.1

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

Description Statistical processing backend for 'ggstatsplot',
this package creates expressions with details from statistical tests.
It can additionally return dataframes with these results, which also
make these functions a more pipe-friendly way to do statistical
analysis. Currently, it supports only the most common types of
statistical tests: parametric, nonparametric, robust, and Bayesian
versions of t-test/ANOVA, correlation analyses, contingency table
analysis, and meta-analysis.

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 (>= 3.6.0)

Imports correlation, dplyr, effectsize, ez, insight (>= 0.11.0),
ipmisc, metafor, metaplus, parameters (>= 0.10.0), performance
(>= 0.6.0), rcompanion, rlang, stats, tidyBF, tidyr, WRS2

Suggests ggplot2, knitr, rmarkdown, spelling, testthat, utils

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.1

Config/testthat/edition 3

Config/testthat/parallel true

Config/testthat/start-first watcher, parallel*

NeedsCompilation no

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

Repository CRAN

Date/Publication 2020-12-01 21:50:03 UTC

R topics documented:

`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)
```

---

expr_anova_bayes *Expression containing Bayesian one-way ANOVA results*

Description

Expression containing Bayesian one-way ANOVA results

Usage

```
expr_anova_bayes(
  data,
  x,
  y,
  subject.id = NULL,
  paired = FALSE,
  bf.prior = 0.707,
  k = 2L,
```

```
  output = "expression",
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. A matrix or tables will **not** be accepted.
`x`	The grouping variable from the dataframe `data`.
`y`	The response (a.k.a. outcome or dependent) variable from the dataframe `data`.
`subject.id`	In case of repeated measures design (`paired = TRUE`, i.e.), this argument specifies the subject or repeated measures id. 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.
`paired`	Logical that decides whether the experimental design is repeated measures/within-subjects or between-subjects. The default is `FALSE`.
`bf.prior`	A number between `0.5` and `2` (default `0.707`), the prior width to use in calculating Bayes factors.
`k`	Number of digits after decimal point (should be an integer) (Default: `k = 2L`).
`output`	If `"expression"`, will return expression with statistical details, while `"dataframe"` will return a dataframe containing the results.
`...`	Additional arguments (currently ignored).

Value

For more details, see- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

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

expr_anova_bayes(
  data = ggplot2::msleep,
  x = vore,
  y = sleep_rem
)
```

`expr_anova_nonparametric` *Making text expression for non-parametric ANOVA.*

Description

Making text expression for non-parametric ANOVA.

Usage

```
expr_anova_nonparametric(
  data,
  x,
  y,
  subject.id = NULL,
  paired = FALSE,
  k = 2L,
  conf.level = 0.95,
  conf.type = "perc",
  nboot = 100L,
  output = "expression",
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. A matrix or tables will **not** be accepted.
`x`	The grouping variable from the dataframe `data`.
`y`	The response (a.k.a. outcome or dependent) variable from the dataframe `data`.
`subject.id`	In case of repeated measures design (`paired = TRUE`, i.e.), this argument specifies the subject or repeated measures id. 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.
`paired`	Decides whether the design is repeated measures or not (Default: `FALSE`).
`k`	Number of digits after decimal point (should be an integer) (Default: `k = 2L`).
`conf.level`	Scalar between 0 and 1. If unspecified, the defaults return 95% confidence/credible intervals (`0.95`).
`conf.type`	A vector of character strings representing the type of intervals required. The value should be any subset of the values `"norm"`, `"basic"`, `"perc"`, `"bca"`. For more, see `?boot::boot.ci`.
`nboot`	Number of bootstrap samples for computing confidence interval for the effect size (Default: `100`).

output	If "expression", will return expression with statistical details, while "dataframe" will return a dataframe containing the results.
...	Additional arguments (currently ignored).

Details

For paired designs, the effect size is Kendall's coefficient of concordance (*W*), while for between-subjects designs, the effect size is epsilon-squared (for more, see ?rcompanion::epsilonSquared and ?rcompanion::kendallW).

Value

For more details, see- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

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

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

# creating the expression
expr_anova_nonparametric(
  data = bugs_long,
  x = condition,
  y = desire,
  paired = TRUE,
  conf.level = 0.99,
  k = 2
)

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

expr_anova_nonparametric(
  data = ggplot2::msleep,
  x = vore,
  y = sleep_rem,
  paired = FALSE,
  conf.level = 0.99,
  conf.type = "perc"
)
```

---

expr_anova_parametric *Expression containing parametric ANOVA results*

Description

The effect sizes and their confidence intervals are computed using `effectsize::eta_squared` and `effectsize::omega_squared` functions.

Usage

```
expr_anova_parametric(
  data,
  x,
  y,
  subject.id = NULL,
  paired = FALSE,
  k = 2L,
  conf.level = 0.95,
  effsize.type = "omega",
  var.equal = FALSE,
  output = "expression",
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. A matrix or tables will **not** be accepted.
`x`	The grouping variable from the dataframe `data`.
`y`	The response (a.k.a. outcome or dependent) variable from the dataframe `data`.
`subject.id`	In case of repeated measures design (`paired = TRUE`, i.e.), this argument specifies the subject or repeated measures id. 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.
`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. If unspecified, the defaults return 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 variances in the samples as equal. If `TRUE`, then a simple F test for the equality of means in a one-way analysis of variance is performed. If `FALSE`, an approximate method of Welch (1951) is used, which generalizes the commonly known 2-sample Welch test to the case of arbitrarily many samples.
`output`	If `"expression"`, will return expression with statistical details, while `"dataframe"` will return a dataframe containing the results.
`...`	Additional arguments (currently ignored).

Value

For more details, see- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

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

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

# to get expression
expr_anova_parametric(
  data = ggplot2::msleep,
  x = vore,
  y = sleep_rem
)

# -------------------- repeated measures ------------------------------

# to get dataframe
expr_anova_parametric(
  data = iris_long,
  x = condition,
  y = value,
  subject.id = id,
  paired = TRUE,
  output = "dataframe"
)
```

---

expr_anova_robust *Expression containing results from heteroscedastic one-way ANOVA for trimmed means*

Description

Expression containing results from heteroscedastic one-way ANOVA for trimmed means

Usage

```
expr_anova_robust(
  data,
  x,
  y,
  subject.id = NULL,
  paired = FALSE,
  k = 2L,
  conf.level = 0.95,
```

```
  tr = 0.1,
  nboot = 100L,
  output = "expression",
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. A matrix or tables will **not** be accepted.
`x`	The grouping variable from the dataframe `data`.
`y`	The response (a.k.a. outcome or dependent) variable from the dataframe `data`.
`subject.id`	In case of repeated measures design (`paired = TRUE`, i.e.), this argument specifies the subject or repeated measures id. 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.
`paired`	Decides whether the design is repeated measures or not (Default: `FALSE`).
`k`	Number of digits after decimal point (should be an integer) (Default: `k = 2L`).
`conf.level`	Scalar between 0 and 1. If unspecified, the defaults return 95% confidence/credible intervals (`0.95`).
`tr`	Trim level for the mean when carrying out `robust` tests. If you get error stating "Standard error cannot be computed because of Winsorized variance of 0 (e.g., due to ties). Try to decrease the trimming level.", try to play around with the value of `tr`, which is by default set to `0.1`. Lowering the value might help.
`nboot`	Number of bootstrap samples for computing confidence interval for the effect size (Default: `100`).
`output`	If `"expression"`, will return expression with statistical details, while `"dataframe"` will return a dataframe containing the results.
`...`	Additional arguments (currently ignored).

Value

For more details, see- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

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

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

expr_anova_robust(
  data = ggplot2::midwest,
  x = state,
```

```
  y = percbelowpoverty
)

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

expr_anova_robust(
  data = iris_long,
  x = condition,
  y = value,
  paired = TRUE,
  k = 3
)
```

`expr_contingency_tab` *Making expression for contingency table analysis*

Description

Making expression for contingency table analysis

Usage

```
expr_contingency_tab(
  data,
  x,
  y = NULL,
  counts = NULL,
  paired = FALSE,
  ratio = NULL,
  k = 2L,
  conf.level = 0.95,
  output = "expression",
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. A matrix or tables will **not** be accepted.
`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.
`counts`	A string naming a variable in data containing counts, or `NULL` if each row represents a single observation.
`paired`	Logical indicating whether data came from a within-subjects or repeated measures design study (Default: `FALSE`). If `TRUE`, McNemar's test subtitle will be returned. If `FALSE`, Pearson's chi-square test will be returned.

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. If unspecified, the defaults return 95% confidence/credible intervals (0.95).
output	If "expression", will return expression with statistical details, while "dataframe" will return a dataframe containing the results.
...	Additional arguments (currently ignored).

Value

Expression or a dataframe for contingency analysis (Pearson's chi-square test for independence for between-subjects design or McNemar's test for within-subjects design) or goodness of fit test for a single categorical variable.

References

For more details, see- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

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

# ----------------------- association tests ----------------------------

# without counts data
expr_contingency_tab(
  data = mtcars,
  x = am,
  y = cyl,
  paired = FALSE
)

# ----------------------- goodness of fit tests --------------------------

# with counts
expr_contingency_tab(
  data = as.data.frame(HairEyeColor),
  x = Eye,
  counts = Freq,
  ratio = c(0.2, 0.2, 0.3, 0.3)
)
```

`expr_corr_test` *Making expression for correlation analysis*

Description

Making expression for correlation analysis

Usage

```
expr_corr_test(
  data,
  x,
  y,
  k = 2L,
  conf.level = 0.95,
  beta = 0.1,
  type = "parametric",
  bf.prior = 0.707,
  output = "expression",
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. A matrix or tables will **not** be accepted.
`x`	The column in `data` containing the explanatory variable to be plotted on the x-axis. Can be entered either as a character string (e.g., `"x"`) or as a bare expression (e.g, `x`).
`y`	The column in `data` containing the response (outcome) variable to be plotted on the y-axis. Can be entered either as a character string (e.g., `"y"`) or as a bare expression (e.g, `y`).
`k`	Number of digits after decimal point (should be an integer) (Default: `k = 2L`).
`conf.level`	Scalar between 0 and 1. If unspecified, the defaults return 95% confidence/credible intervals (`0.95`).
`beta`	bending constant (Default: `0.1`). For more, see `WRS2::pbcor()`.
`type`	Type of association between paired samples required (`""parametric"`: Pearson's product moment correlation coefficient" or `""nonparametric"`: Spearman's rho" or `""robust"`: percentage bend correlation coefficient" or `""bayes"`: Bayes Factor for Pearson's *r*"). Corresponding abbreviations are also accepted: `"p"` (for parametric/pearson), `"np"` (nonparametric/spearman), `"r"` (robust), `"bf"` (for bayes factor), resp.
`bf.prior`	A number between `0.5` and `2` (default `0.707`), the prior width to use in calculating Bayes factors.

output If "expression", will return expression with statistical details, while "dataframe" will return a dataframe containing the results.

... Arguments passed on to bf_extractor

conf.level Confidence/Credible Interval (CI) level. Default to 0.95 (95%).

centrality The point-estimates (centrality indices) to compute. Character (vector) or list with one or more of these options: "median", "mean", "MAP" or "all".

conf.method The type of index used for Credible Interval. Can be "hdi" (default), "eti", or "si" (see si(), hdi(), eti() functions from bayestestR package).

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

top.text Text to display on top of the Bayes Factor message. This is mostly relevant in the context of ggstatsplot functions.

output If "expression", will return expression with statistical details, while "dataframe" will return a dataframe containing the results.

Value

Expression containing results from correlation test with confidence intervals for the correlation coefficient estimate. Results are extracted via correlation::correlation.

References

For more details, see- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

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

# without changing defaults
expr_corr_test(
  data = ggplot2::midwest,
  x = area,
  y = percblack,
  type = "parametric"
)

# changing defaults
expr_corr_test(
  data = ggplot2::midwest,
  x = area,
  y = percblack,
  beta = 0.2,
  type = "robust"
)
```

expr_meta_random *Making expression for random-effects meta-analysis*

Description

Making expression for random-effects meta-analysis

Usage

```
expr_meta_random(
  data,
  type = "parametric",
  metaBMA.args = list(),
  random = "mixture",
  k = 2L,
  conf.level = 0.95,
  caption = NULL,
  output = "expression",
  ...
)
```

Arguments

data	A dataframe. It **must** contain columns named `estimate` (effect sizes or outcomes) and `std.error` (corresponding standard errors). These two columns will be used for `yi` and `sei` arguments in `metafor::rma` (for parametric analysis) or `metaplus::metaplus` (for robust analysis).
type	Type of statistic expected (`"parametric"` or `"nonparametric"` or `"robust"` or `"bayes"`).Corresponding abbreviations are also accepted: `"p"` (for parametric), `"np"` (nonparametric), `"r"` (robust), or `"bf"`resp.
metaBMA.args	A list of additional arguments to be passed to `metaBMA::meta_random`.
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. If unspecified, the defaults return 95% confidence/credible intervals (`0.95`).
caption	Text to display as caption. This argument is relevant only when `output = "caption"`.
output	If `"expression"`, will return expression with statistical details, while `"dataframe"` will return a dataframe containing the results.
...	Additional arguments passed to the respective meta-analysis function.

Details

This analysis is carried out using-

- parametric: `metafor::rma`
- robust: `metaplus::metaplus`
- Bayesian: `metaBMA::meta_random`

Examples

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

# renaming to what `statsExpressions` expects
df <- dplyr::rename(mag, estimate = yi, std.error = sei)

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

# creating expression
expr_meta_random(data = df, k = 3)

# ---------------------- random --------------------------------------------

# creating expression
expr_meta_random(
  data = df,
  type = "random",
  random = "normal",
  output = "dataframe"
)

# ---------------------- Bayes Factor --------------------------------------

# making subtitle
expr_meta_random(
  data = df,
  type = "bayes",
  k = 3,
  # additional arguments given to `metaBMA`
  metaBMA.args = list(
    iter = 5000,
    summarize = "integrate",
    control = list(adapt_delta = 0.99, max_treedepth = 15)
  )
)
```

`expr_template` *Template for subtitles with statistical details for tests*

Description

Template for subtitles with statistical details for tests

Usage

```
expr_template(
  no.parameters,
  statistic.text,
  stats.df,
  effsize.text,
  n,
  conf.level = 0.95,
  k = 2L,
  k.parameter = 0L,
  k.parameter2 = 0L,
  n.text = NULL,
  ...
)
```

Arguments

`no.parameters`	An integer that specifies that the number of parameters for the statistical test. Can be `0` for non-parametric tests, `1` for tests based on *t*-statistic or chi-squared statistic, `2` for tests based on *F*-statistic.
`statistic.text`	A character that specifies the relevant test statistic. For example, for tests with *t*-statistic, `statistic.text = "t"`. If you want to use plotmath, you will have to quote the argument (e.g., `quote(italic("t"))`).
`stats.df`	A dataframe containing details from the statistical analysis and should contain some of the the following columns: • *statistic*: the numeric value of a statistic. • *parameter*: the numeric value of a parameter being modeled (often degrees of freedom for the test); note that if `no.parameters = 0L` (e.g., for non-parametric tests), this column will be irrelevant. • *parameter1*, *parameter2* 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. • *estimate*: estimated value of the effect size. • *conf.low*: lower bound for effect size estimate. • *conf.high*: upper bound for effect size estimate.
`effsize.text`	A character that specifies the relevant effect size. For example, for Cohen's *d* statistic, `effsize.text = "d"`. If you want to use plotmath, you will have to quote the argument (e.g., `quote(italic("d"))`).

n	An integer specifying the sample size used for the test.
conf.level	Scalar between 0 and 1. If unspecified, the defaults return 95% confidence/credible intervals (0.95).
k	Number of digits after decimal point (should be an integer) (Default: k = 2L).
k.parameter, k.parameter2	Number of decimal places to display for the parameters (default: 0).
n.text	A character that specifies the design, which will determine what the n stands for. For example, for repeated measures, this can be quote(italic("n")["pairs"]), while for independent subjects design this can be quote(italic("n")["obs"]). If NULL, defaults to generic quote(italic("n")).
...	Currently ignored.

Examples

```
set.seed(123)

# creating a dataframe with stats results
stats_df <-
  cbind.data.frame(
    statistic = 5.494,
    parameter = 29.234,
    p.value = 0.00001,
    estimate = -1.980,
    conf.low = -2.873,
    conf.high = -1.088
  )

# subtitle for *t*-statistic with Cohen's *d* as effect size
statsExpressions::expr_template(
  no.parameters = 1L,
  stats.df = stats_df,
  statistic.text = quote(italic("t")),
  effsize.text = quote(italic("d")),
  n = 32L,
  conf.level = 0.95,
  k = 3L,
  k.parameter = 3L
)
```

---

expr_t_bayes *Making expression containing Bayesian* t*-test results*

Description

Making expression containing Bayesian *t*-test results

Usage

```
expr_t_bayes(
  data,
  x,
  y,
  subject.id = NULL,
  paired = FALSE,
  k = 2L,
  bf.prior = 0.707,
  output = "expression",
  ...
)
```

Arguments

data	A dataframe (or a tibble) from which variables specified are to be taken. A matrix or tables will **not** be accepted.
x	The grouping variable from the dataframe `data`.
y	The response (a.k.a. outcome or dependent) variable from the dataframe `data`.
subject.id	In case of repeated measures design (`paired = TRUE`, i.e.), this argument specifies the subject or repeated measures id. 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.
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`).
bf.prior	A number between `0.5` and `2` (default `0.707`), the prior width to use in calculating Bayes factors.
output	If `"expression"`, will return expression with statistical details, while `"dataframe"` will return a dataframe containing the results.
...	Additional arguments (currently ignored).

References

For more details, see- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

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

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

expr_t_bayes(
```

```
  data = mtcars,
  x = am,
  y = wt,
  paired = FALSE
)

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

expr_t_bayes(
  data = dplyr::filter(bugs_long, condition %in% c("LDLF", "LDHF")),
  x = condition,
  y = desire,
  paired = TRUE,
  subject.id = subject
)
```

`expr_t_nonparametric` *Making expression for Mann-Whitney* U*-test/Wilcoxon test results*

Description

Making expression for Mann-Whitney *U*-test/Wilcoxon test results

Usage

```
expr_t_nonparametric(
  data,
  x,
  y,
  subject.id = NULL,
  paired = FALSE,
  k = 2L,
  conf.level = 0.95,
  conf.type = "norm",
  nboot = 100,
  output = "expression",
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. A matrix or tables will **not** be accepted.
`x`	The grouping variable from the dataframe `data`.
`y`	The response (a.k.a. outcome or dependent) variable from the dataframe `data`.

`subject.id`	In case of repeated measures design (`paired = TRUE`, i.e.), this argument specifies the subject or repeated measures id. 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.
`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. If unspecified, the defaults return 95% confidence/credible intervals (`0.95`).
`conf.type`	A vector of character strings representing the type of intervals required. The value should be any subset of the values `"norm"`, `"basic"`, `"perc"`, `"bca"`. For more, see `?boot::boot.ci`.
`nboot`	Number of bootstrap samples for computing confidence interval for the effect size (Default: `100`).
`output`	If `"expression"`, will return expression with statistical details, while `"dataframe"` will return a dataframe containing the results.
`...`	Additional arguments (currently ignored).

Details

For the two independent samples case, the Mann-Whitney U-test is calculated and W is reported from *stats::wilcox.test*. For the paired samples case the Wilcoxon signed rank test is run and V is reported.

Since there is no single commonly accepted method for reporting effect size for these tests we are computing and reporting r (computed as $Z/\sqrt{N}$) along with the confidence intervals associated with the estimate. Note that N here corresponds to total *sample size* for independent/between-subjects designs, and to total number of *pairs* (and **not** *observations*) for repeated measures/within-subjects designs.

Note: The *stats::wilcox.test* function does not follow the same convention as *stats::t.test*. The sign of the V test statistic will always be positive since it is **the sum of the positive signed ranks**. Therefore, V will vary in magnitude but not significance based solely on the order of the grouping variable. Consider manually reordering your factor levels if appropriate as shown in the second example below.

References

For more details, see- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

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

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

```
expr_t_nonparametric(
  data = sleep,
  x = group,
  y = extra
)

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

expr_t_nonparametric(
  data = VR_dilemma,
  x = modality,
  y = score,
  paired = TRUE,
  subject.id = id
)
```

`expr_t_onesample` *Expression for one sample* t*-test and its non-parametric and robust equivalents*

Description

Expression for one sample *t*-test and its non-parametric and robust equivalents

Usage

```
expr_t_onesample(
  data,
  x,
  type = "parametric",
  test.value = 0,
  k = 2L,
  conf.level = 0.95,
  conf.type = "norm",
  bf.prior = 0.707,
  robust.estimator = "onestep",
  effsize.type = "g",
  nboot = 100L,
  output = "expression",
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. A matrix or tables will **not** be accepted.
`x`	A numeric variable from the dataframe `data`.

type	Type of statistic expected ("parametric" or "nonparametric" or "robust" or "bayes").Corresponding abbreviations are also accepted: "p" (for parametric), "np" (nonparametric), "r" (robust), or "bf"resp.
test.value	A number specifying the value of the null hypothesis (Default: 0).
k	Number of digits after decimal point (should be an integer) (Default: k = 2L).
conf.level	Scalar between 0 and 1. If unspecified, the defaults return 95% confidence/credible intervals (0.95).
conf.type	A vector of character strings representing the type of intervals required. The value should be any subset of the values "norm", "basic", "perc", "bca". For more, see ?boot::boot.ci.
bf.prior	A number between 0.5 and 2 (default 0.707), the prior width to use in calculating Bayes factors.
robust.estimator	If type = "robust", a robust estimator to be used ("onestep" (Default), "mom", or "median"). For more, see ?WRS2::onesampb.
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*).
nboot	Number of bootstrap samples for computing confidence interval for the effect size (Default: 100).
output	If "expression", will return expression with statistical details, while "dataframe" will return a dataframe containing the results.
...	Additional arguments passed to tidyBF::bf_ttest.

Value

Expression containing results from a one-sample test. The exact test and the effect size details contained will be dependent on the type argument.

References

For more details, see- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

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

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

expr_t_onesample(
  data = ggplot2::msleep,
  x = brainwt,
  test.value = 0.275,
  type = "parametric"
```

```
)

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

expr_t_onesample(
  data = ggplot2::msleep,
  x = brainwt,
  test.value = 0.275,
  type = "nonparametric"
)

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

expr_t_onesample(
  data = ggplot2::msleep,
  x = brainwt,
  test.value = 0.275,
  type = "robust"
)

# ----------------------- Bayes Factor ------------------------------------

expr_t_onesample(
  data = ggplot2::msleep,
  x = brainwt,
  test.value = 0.275,
  type = "bayes",
  bf.prior = 0.8
)
```

`expr_t_parametric` *Making expression containing* t*-test results*

Description

Making expression containing *t*-test results

Usage

```
expr_t_parametric(
  data,
  x,
  y,
  subject.id = NULL,
  paired = FALSE,
  k = 2L,
  conf.level = 0.95,
  effsize.type = "g",
```

```
  var.equal = FALSE,
  output = "expression",
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. A matrix or tables will **not** be accepted.
`x`	The grouping variable from the dataframe `data`.
`y`	The response (a.k.a. outcome or dependent) variable from the dataframe `data`.
`subject.id`	In case of repeated measures design (`paired = TRUE`, i.e.), this argument specifies the subject or repeated measures id. 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.
`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. If unspecified, the defaults return 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 variances in the samples as equal. If `TRUE`, then a simple F test for the equality of means in a one-way analysis of variance is performed. If `FALSE`, an approximate method of Welch (1951) is used, which generalizes the commonly known 2-sample Welch test to the case of arbitrarily many samples.
`output`	If `"expression"`, will return expression with statistical details, while `"dataframe"` will return a dataframe containing the results.
`...`	Additional arguments (currently ignored).

Details

Cohen's *d* is calculated in the traditional fashion as the difference between means or mean minus *mu* divided by the estimated standardized deviation. By default Hedge's correction is applied (*N*-3)/(*N*-2.25) to produce *g*. For independent samples *t*-test, there are two possibilities implemented. If the *t*-test did not make a homogeneity of variance assumption, (the Welch test), the variance term will mirror the Welch test, otherwise a pooled and weighted estimate is used. If a paired samples *t*-test was requested, then effect size desired is based on the standard deviation of the differences.

The computation of the confidence intervals defaults to a use of non-central Student-*t* distributions.

When computing confidence intervals the variance of the effect size *d* or *g* is computed using the conversion formula reported in Cooper et al. (2009)

- `((n1+n2)/(n1*n2) + .5*d^2/df) * ((n1+n2)/df)` (independent samples)
- `sqrt(((1 / n) + (d^2 / n)) * 2 * (1 -r))` (paired case)

Value

Expression containing details from results of a two-sample test and effect size plus confidence intervals.

References

For more details, see- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

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

# creating a smaller dataset
msleep_short <- dplyr::filter(ggplot2::msleep, vore %in% c("carni", "herbi"))

# with defaults
expr_t_parametric(
  data = msleep_short,
  x = vore,
  y = sleep_rem
)

# changing defaults (getting expression as output)
expr_t_parametric(
  data = msleep_short,
  x = vore,
  y = sleep_rem,
  var.equal = TRUE,
  effsize.type = "d"
)
```

---

expr_t_robust *Expression containing results from a robust* t*-test*

Description

Expression containing results from a robust *t*-test

Usage

```
expr_t_robust(
  data,
  x,
  y,
  subject.id = NULL,
  paired = FALSE,
```

```
  k = 2L,
  conf.level = 0.95,
  tr = 0.1,
  nboot = 100,
  output = "expression",
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. A matrix or tables will **not** be accepted.
`x`	The grouping variable from the dataframe `data`.
`y`	The response (a.k.a. outcome or dependent) variable from the dataframe `data`.
`subject.id`	In case of repeated measures design (`paired = TRUE`, i.e.), this argument specifies the subject or repeated measures id. 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.
`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. If unspecified, the defaults return 95% confidence/credible intervals (`0.95`).
`tr`	Trim level for the mean when carrying out `robust` tests. If you get error stating "Standard error cannot be computed because of Winsorized variance of 0 (e.g., due to ties). Try to decrease the trimming level.", try to play around with the value of `tr`, which is by default set to `0.1`. Lowering the value might help.
`nboot`	Number of bootstrap samples for computing confidence interval for the effect size (Default: `100`).
`output`	If `"expression"`, will return expression with statistical details, while `"dataframe"` will return a dataframe containing the results.
`...`	Additional arguments (currently ignored).

References

For more details, see- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

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

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

```
# with defaults
expr_t_robust(
  data = sleep,
  x = group,
  y = extra
)

# within-subjects design ----------------------------------------------
expr_t_robust(
  data = dplyr::filter(bugs_long, condition %in% c("LDLF", "LDHF")),
  x = condition,
  y = desire,
  paired = TRUE,
  subject.id = subject
)
```

`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)
```

`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)
```

tidy_model_parameters *Convert* parameters *output to* tidymodels *convention*

Description

Convert parameters output to tidymodels convention

Usage

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

Arguments

model	Statistical Model.
...	Arguments passed to or from other methods. Non-documented arguments are digits, p_digits and ci_digits to set the number of digits for the output. See 'Examples' in model_parameters.default.

Examples

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

tidy_model_performance *Convert* performance *output to* tidymodels *convention*

Description

Convert performance output to tidymodels convention

Usage

```
tidy_model_performance(model, ...)
```

Arguments

model	Statistical model.
...	Arguments passed to or from other methods, resp. for compare_performance(), one or multiple model objects (also of different classes).

Examples

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

VR_dilemma *Virtual reality moral dilemmas.*

Description

Virtual reality moral dilemmas.

Usage

```
VR_dilemma
```

Format

A data frame with 68 rows and 4 variables

- id. Dummy identity number for each participant.
- order. The order in which the participants completed the two sessions: `"text_first"` (`0`) or `"text_second"` (`1`).
- modality. Describes how the moral dilemmas were presented to the participants: either in text format (`"text"`) or in Virtual Reality (`"vr"`).
- score. Proportion of "utilitarian" decisions. In other words, of the 4 decisions, how many affirmative were responses. Range: 0 (all utilitarian) - 1 (none utilitarian).

Details

Dataset from a study where participants completed identical moral dilemmas in two different sessions held on separate days: in one session, they read text description of the scenario, while in another session they completed the same scenarios in Virtual Reality (videos: https://www.youtube.com/watch?v=ebdU3HhhYs8). The study investigated if there was a discrepancy between how people judged the same scenarios while reading them in text versus experiencing them in virtual reality.

Source

https://psyarxiv.com/ry3ap/

Examples

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

Index