Package 'statsExpressions'

May 30, 2021

Type Package

Title Tidy Dataframes and Expressions with Statistical Details

Version 1.1.0

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'.

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 BayesFactor (>= 0.9.12-4.2), correlation (>= 0.6.1), dplyr, effectsize (>= 0.4.5), insight (>= 0.14.1), ipmisc, parameters (>= 0.14.0), performance (>= 0.7.2), purrr, rlang, stats, tidyr, WRS2 (>= 1.1-1)

Suggests afex, ggplot2, knitr, metaBMA, metafor, metaplus, rmarkdown, spelling, testthat, utils

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.1.9001

Config/testthat/edition 3

Config/testthat/parallel true

NeedsCompilation no

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

Repository CRAN

Date/Publication 2021-05-30 04:30:02 UTC

R topics documented:

`statsExpressions-package`

statsExpressions: Tidy Dataframes and Expressions with Statistical Details

---

Description

`statsExpressions` package produces tidy 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` package.

For more documentation, see the dedicated Website.

Details

`statsExpressions`

Author(s)

Maintainer: Indrajeet Patil <patilindrajeet.science@gmail.com> (ORCID) (@patilindrajeets) [copyright holder]

See Also

Useful links:

- https://indrajeetpatil.github.io/statsExpressions/
- https://github.com/IndrajeetPatil/statsExpressions
- Report bugs at https://github.com/IndrajeetPatil/statsExpressions/issues

---

bf_extractor *Extract Bayes Factors from* BayesFactor *model object.*

Description

Extract Bayes Factors from BayesFactor model object.

Usage

```
bf_extractor(bf.object, conf.level = 0.95, k = 2L, top.text = NULL, ...)
```

Arguments

bf.object	An object from BayesFactor package.
conf.level	Confidence/Credible Interval (CI) level. Default to 0.95 (95%).
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.
...	Additional arguments passed to parameters::model_parameters.BFBayesFactor().

Note

Important: don't enter 1/bf.object to extract results for null hypothesis; doing so will return wrong results.

Examples

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

# creating a `BayesFactor` object
bf_obj <-
```

```
  BayesFactor::anovaBF(
    formula = Sepal.Length ~ Species,
    data = iris,
    progress = FALSE
  )

# extracting Bayes Factors in a dataframe
bf_extractor(bf_obj)
```

---

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)
```

`contingency_table` *Contingency table analyses*

Description

A dataframe containing results from for contingency table analysis or goodness of fit test.

To see details about functions which are internally used to carry out these analyses, see the following vignette- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

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,
  top.text = NULL,
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix,table, array, etc.) 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.
`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	A string naming a 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	Confidence/Credible Interval (CI) level. Default to 0.95 (95%).
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.
top.text	Text to display on top of the Bayes Factor message. This is mostly relevant in the context of ggstatsplot functions.
...	Additional arguments (currently ignored).

Examples

```
# for reproducibility
set.seed(123)
library(statsExpressions)
options(tibble.width = Inf, pillar.bold = TRUE, pillar.neg = TRUE)

# ------------------------ non-Bayesian ----------------------------

# association test
contingency_table(
  data = mtcars,
  x = am,
  y = cyl,
  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 = cyl,
  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

A dataframe containing results from correlation test with confidence intervals for the correlation coefficient estimate.

Usage

```
corr_test(
  data,
  x,
  y,
  type = "parametric",
  k = 2L,
  conf.level = 0.95,
  tr = 0.2,
  bf.prior = 0.707,
  top.text = NULL,
```

```
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix,table, array, etc.) will **not** be accepted.
`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`. If unspecified, the defaults return 95% confidence/credible intervals (`0.95`).
`tr`	Trim level for the mean when carrying out `robust` tests. In case of an error, try reducing the value of `tr`, which is by default set to `0.2`. Lowering the value might help.
`bf.prior`	A number between `0.5` and `2` (default `0.707`), the prior width to use in calculating Bayes factors and posterior estimates. In addition to numeric arguments, several named values are also recognized: `"medium"`, `"wide"`, and `"ultrawide"`, corresponding to *r* scale values of 1/2, sqrt(2)/2, and 1, respectively. In case of an ANOVA, this value corresponds to scale for fixed effects.
`top.text`	Text to display on top of the Bayes Factor message. This is mostly relevant in the context of `ggstatsplot` functions.
`...`	Additional arguments (currently ignored).

References

To see details about functions which are internally used to carry out these analyses, see the following vignette- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Examples

```
# for reproducibility
set.seed(123)
library(statsExpressions)
options(tibble.width = Inf, pillar.bold = TRUE, pillar.neg = TRUE)
```

```
# without changing defaults
corr_test(
  data = ggplot2::midwest,
  x = area,
  y = percblack
)

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

`expr_template` *Template for expressions with statistical details*

Description

Creates an expression from a dataframe containing statistical details. Ideally, this dataframe 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

```
expr_template(
  data,
  no.parameters = 0L,
  bayesian = FALSE,
  statistic.text = NULL,
  effsize.text = NULL,
  top.text = NULL,
  prior.distribution = NULL,
  prior.type = NULL,
  n = NULL,
  n.text = NULL,
  paired = FALSE,
  conf.method = "HDI",
  k = 2L,
  k.df = 0L,
  k.df.error = 0L,
  ...
)
```

Arguments

`data`	A dataframe 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 `no.parameters = 0L` (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. • *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`). • *method*: method describing the test carried out.
`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.
`bayesian`	Is this Bayesian analysis? Defaults to `FALSE`. The template is slightly different for Bayesian analysis.
`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.
`top.text`	Text to display on top of the Bayes Factor message. This is mostly relevant in the context of `ggstatsplot` functions.
`prior.distribution`	A character that specifies the prior type.
`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. If `NULL`, defaults to `quote(italic("n")["pairs"])` if `paired = TRUE`, and to `quote(italic("n")["obs"])` if `paired = FALSE`.
`paired`	Logical that decides whether the experimental design is repeated measures/within-subjects or between-subjects. The default is `FALSE`.
`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`).
`k.df, k.df.error`	Number of decimal places to display for the parameters (default: `0`).
`...`	Currently ignored.

Examples

```
set.seed(123)

# creating a dataframe 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
  )

# expression for *t*-statistic with Cohen's *d* as effect size
# note that the plotmath expressions need to be quoted
statsExpressions::expr_template(
  no.parameters = 1L,
  data = stats_df,
  statistic.text = quote(italic("t")),
  effsize.text = quote(italic("d")),
  n = 32L,
  k = 3L,
  k.df = 3L
)
```

---

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)
```

---

meta_analysis *Random-effects meta-analyses*

Description

A dataframe containing results from random-effects meta-analysis.

To see details about functions which are internally used to carry out these analyses, see the following vignette- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

Usage

```
meta_analysis(
  data,
  type = "parametric",
  random = "mixture",
  k = 2L,
  conf.level = 0.95,
  top.text = NULL,
  ...
)
```

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: • 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`	Confidence/Credible Interval (CI) level. Default to `0.95` (95%).
`top.text`	Text to display on top of the Bayes Factor message. This is mostly relevant in the context of `ggstatsplot` functions.
`...`	Additional arguments passed to the respective meta-analysis function.

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

```
# run examples only if the needed packages are available
if (all(unlist(lapply(
  c("metaplus", "metafor", "metaBMA"), # needed packages
  require,
  character.only = TRUE,
  quietly = TRUE,
  warn.conflicts = FALSE
)))) {
  # note that the `print` calls below are not necessary for you to write
  # they are in the documentation so that the website renders them

  # setup
  set.seed(123)
  library(statsExpressions)
  options(tibble.width = Inf, pillar.bold = TRUE, pillar.neg = TRUE)

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

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

  print(meta_analysis(data = df))

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

  print(meta_analysis(
    data = df,
    type = "random",
    random = "normal"
  ))
```

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

  meta_analysis(
    data = df,
    type = "bayes",

    # additional arguments given to `metaBMA`
    iter = 5000,
    summarize = "integrate",
    control = list(adapt_delta = 0.99, max_treedepth = 15)
  )
}
```

`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

A dataframe containing results for one-way ANOVA.

To see details about functions which are internally used to carry out these analyses, see the following vignette- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

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,
  top.text = NULL,
  ...
)
```

Arguments

data	A dataframe (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix,table, array, etc.) will **not** be accepted.
x	The grouping (or independent) variable from the dataframe `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 the dataframe `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`. 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 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`).
`top.text`	Text to display on top of the Bayes Factor message. This is mostly relevant in the context of `ggstatsplot` functions.
`...`	Additional arguments (currently ignored).

Note

To carry out Bayesian posterior estimation for ANOVA designs, you will need to install the development version of `BayesFactor` (0.9.12-4.3). You can download it by running: `remotes::install_github("richarddmorey/`

Examples

```
# for reproducibility
set.seed(123)
library(statsExpressions)
options(tibble.width = Inf, pillar.bold = TRUE, pillar.neg = TRUE)

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

# between-subjects
oneway_anova(
  data = ggplot2::msleep,
  x = vore,
  y = sleep_rem
)

if (require("afex", quietly = TRUE)) {
  # within-subjects design
  oneway_anova(
    data = iris_long,
    x = condition,
    y = value,
    subject.id = id,
    paired = TRUE
  )
}

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

# between-subjects
oneway_anova(
  data = ggplot2::msleep,
  x = vore,
  y = sleep_rem,
  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 = ggplot2::msleep,
```

```
  x = vore,
  y = sleep_rem,
  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 = ggplot2::msleep,
  x = vore,
  y = sleep_rem,
  type = "bayes"
)

# within-subjects design
# needs `BayesFactor 0.9.12-4.3` or above
if (utils::packageVersion("BayesFactor") >= package_version("0.9.12-4.3")) {
  oneway_anova(
    data = iris_long,
    x = condition,
    y = value,
    subject.id = id,
    paired = TRUE,
    type = "bayes"
  )
}
```

`one_sample_test` *One-sample tests*

Description

A dataframe containing results from a one-sample test.

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",
  top.text = NULL,
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix,table, array, etc.) will **not** be accepted.
`x`	A numeric variable from the dataframe `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`	Confidence/Credible Interval (CI) level. Default to `0.95` (95%).
`tr`	Trim level for the mean when carrying out `robust` tests. In case of an error, try reducing the value of `tr`, which is by default set to `0.2`. Lowering the value might help.
`bf.prior`	A number between `0.5` and `2` (default `0.707`), the prior width to use in calculating Bayes factors and posterior estimates. In addition to numeric arguments, several named values are also recognized: `"medium"`, `"wide"`, and `"ultrawide"`, corresponding to *r* scale values of 1/2, sqrt(2)/2, and 1, respectively. In case of an ANOVA, this value corresponds to scale for fixed effects.
`effsize.type`	Type of effect size needed for *parametric* tests. The argument can be `"d"` (for Cohen's *d*) or `"g"` (for Hedge's *g*).
`top.text`	Text to display on top of the Bayes Factor message. This is mostly relevant in the context of `ggstatsplot` functions.
`...`	Currently ignored.

Details

The exact test and the effect size details contained will depend on the `type` argument.

Internal function `.f` used to carry out statistical test:

- **parametric**: `stats::t.test`
- **nonparametric**: `stats::wilcox.test`
- **robust**: `trimcibt` (custom)
- **bayes**: `BayesFactor::ttestBF`

Internal function `.f.es` used to compute effect size:

- **parametric**: `effectsize::cohens_d`, `effectsize::hedges_g`
- **nonparametric**: `effectsize::rank_biserial`
- **robust**: `trimcibt` (custom)
- **bayes**: `bayestestR::describe_posterior`

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

Examples

```
# for reproducibility
set.seed(123)
library(statsExpressions)
options(tibble.width = Inf, pillar.bold = TRUE, pillar.neg = TRUE)

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

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

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

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

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

one_sample_test(
  data = ggplot2::msleep,
  x = brainwt,
```

```
  test.value = 0.275,
  type = "robust"
)

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

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

---

tidy_model_effectsize *Convert* effectsize *package output to* tidyverse *conventions*

Description

Convert effectsize package output to tidyverse conventions

Usage

```
tidy_model_effectsize(data)
```

Arguments

data	Dataframe returned by effectsize functions.

Examples

```
df <- effectsize::cohens_d(sleep$extra, sleep$group)
tidy_model_effectsize(df)
```

---

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

A dataframe containing results from a two-sample test and effect size plus confidence intervals.

To see details about functions which are internally used to carry out these analyses, see the following vignette- https://indrajeetpatil.github.io/statsExpressions/articles/stats_details.html

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,
  top.text = NULL,
  ...
)
```

Arguments

`data`	A dataframe (or a tibble) from which variables specified are to be taken. Other data types (e.g., matrix,table, array, etc.) will **not** be accepted.
`x`	The grouping (or independent) variable from the dataframe `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 the dataframe `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`.
`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`	Confidence/Credible Interval (CI) level. Default to `0.95` (95%).
`effsize.type`	Type of effect size needed for *parametric* tests. The argument can be `"d"` (for Cohen's *d*) or `"g"` (for Hedge's *g*).
`var.equal`	a logical variable indicating whether to treat the two variances as being equal. If `TRUE` then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.
`bf.prior`	A number between `0.5` and `2` (default `0.707`), the prior width to use in calculating Bayes factors and posterior estimates. In addition to numeric arguments, several named values are also recognized: `"medium"`, `"wide"`, and `"ultrawide"`, corresponding to *r* scale values of 1/2, sqrt(2)/2, and 1, respectively. In case of an ANOVA, this value corresponds to scale for fixed effects.
`tr`	Trim level for the mean when carrying out `robust` tests. In case of an error, try reducing the value of `tr`, which is by default set to `0.2`. Lowering the value might help.

nboot	Number of bootstrap samples for computing confidence interval for the effect size (Default: 100L).
top.text	Text to display on top of the Bayes Factor message. This is mostly relevant in the context of ggstatsplot functions.
...	Currently ignored.

Examples

```
# for reproducibility
set.seed(123)
library(statsExpressions)
options(tibble.width = Inf, pillar.bold = TRUE, pillar.neg = TRUE)

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

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

# within-subjects design
two_sample_test(
  data = VR_dilemma,
  x = modality,
  y = score,
  paired = TRUE,
  subject.id = id,
  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 = VR_dilemma,
  x = modality,
  y = score,
  paired = TRUE,
  subject.id = id,
  type = "np"
```

```
)

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

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

# within-subjects design
two_sample_test(
  data = VR_dilemma,
  x = modality,
  y = score,
  paired = TRUE,
  subject.id = id,
  type = "r"
)

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

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

# within-subjects design
two_sample_test(
  data = VR_dilemma,
  x = modality,
  y = score,
  paired = TRUE,
  subject.id = id,
  type = "bayes"
)
```

`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