Package ‘neatStats’

June 23, 2019

Title An R Package for Neat and Painless Statistical Reporting

Version 0.1.0

Date 2019-06-22

Description This package focuses on user-friendly, clear and simple statistics, primarily for publication in psychological science. The main functions are wrappers for other packages, but there are various additions as well. Every basic step from data aggregation to reportable printed statistics is covered.

URL https://github.com/gasparl/neatstats

Depends R (>= 3.4.0)

Imports bayestestR, pROC, MBESS, ez, BayesFactor, Exact, rstudioapi

License BSD_2_clause + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

R topics documented:

anova_neat *Comparison of Multiple Means: ANOVA*

Description

Analysis of variance (ANOVA) F-test results with appropriate Welch's and epsilon corrections where applicable (unless specified otherwise), including partial eta squared effect sizes with confidence intervals (CIs), and inclusion Bayes factor based on matched models (BFs).

Usage

```
anova_neat(data_per_subject, values, between_vars = NULL,
  within_ids = NULL, ci = 0.9, bf_added = TRUE,
  test_title = "--- neat ANOVA ---", welch = TRUE, e_correction = "")
```

Arguments

`data_per_subject`	Data frame or name of data frame as string. Should contain all values (measurements/observations) in a single row per each subject.
`values`	String; column name or names. Multiple column names are also to be given as a single string, separated by commas (e.g., `values = 'var1,var2,var3'`). (Spaces are ignored.) Each such column should contain a single dependent variable. This means, to test repeated (within-subject) measurements, each specified column should contain one measurement.
`between_vars`	String; column name or names. Multiple column names are also to be given as a single string, separated by commas (e.g., `values = 'grouping1,grouping2'`). (Spaces are ignored.) Each such column should contain a single between-subject independent variable (representing a between-subject factor).
`within_ids`	String (or number); optionally specifies one-sided tests (t and BF): either "1" (`var1` mean expected to be greater than `var2` mean) or "2" (`var2` mean expected to be greater than `var1` mean). If left empty, the test is two-sided.
`ci`	Numeric; confidence level for returned CIs. (Otherwise 90 default; Lakens, 2014; Steiger, 2004.)
`bf_added`	Logical. If `TRUE` (default), inclusion Bayes factor is calculated and displayed. (Note: with multiple factors and/or larger dataset, the calculation can take considerable time.)
`test_title`	String, "— neat ANOVA —" by default. Simply displayed in printing preceding the statistics.
`welch`	If FALSE, calculates via `ez::ezANOVA` in case of a single factor (one-way) between-subject design (i.e., same as in case of every other design). Otherwise (default), calculates Welch's ANOVA via `stats::oneway.test` in such cases (one-way between-subject).
`e_correction`	String: `'gg'`, `'hf'`, `'none'`, or empty. If set to `'gg'`, Greenhouse-Geisser correction is applied in case of repeated measures (regardless of violation of sphericity). If set to `'hf'`, Huynh-Feldt correction is applied. If set to `'none'`, no correction is applied. Otherwise (e.g. if left empty, `''`, as per default),

Greenhouse-Geisser correction is applied when Mauchly's sphericity test is significant and the Greenhouse-Geisser epsilon is not larger than `.75`, while Huynh-Feldt correction is applied when Mauchly's sphericity test is significant and the Greenhouse-Geisser epsilon is larger than `.75` (see Girden, 1992).

Details

The Bayes factor (BF) is always calculated with the default `rscaleFixed` of `0.5` (`"medium"`) and `rscaleRandom` r-scale of `1` (`"nuisance"`). BF supporting null hypothesis is denoted as BF01, while that supporting alternative hypothesis is denoted as BF10. When the BF is smaller than 1 (i.e., supports null hypothesis), the reciprocal is calculated (hence, BF10 = BF, but BF01 = 1/BF). When the BF is greater than or equal to 10000, scientific (exponential) form is reported for readability. (The original full BF number is available in the returned named vector as `bf`.)

Levene's test is returned for between-subject design without Welch's correction, while Mauchly's sphericity test is returned for repeated measures with more than two levels. If Mauchly's test is significant, epsilon correction may be applied (see the `e_correction` parameter). If Levene's test is significant and the data is unbalanced (unequal group sample sizes), you should either consider the results respectively or choose a different test.

Value

Prints ANOVA statistics (including, for each model, F-test with partial eta squared and its CI, and BF, as specified via the corresponding parameters) in APA style. Furthermore, when assigned, returns a list with up to three elements. First, `'stat_list'`, a list of named vectors per each effect (main or interaction). Each vector contains the following elements: `F` (F value), `p` (p value), `petas` (partial eta squared), `epsilon` (epsilon used for correction), and `bf` (inclusion BF; when `bf_added` is not `FALSE`). Second, the `ezANOVA` object, named `ez_anova` (calculated even when `oneway.test` is printed). Third, when `bf_added` is not `FALSE`, the `anovaBF` object, named `bf_models`; including all models on which the inclusion BFs are based.

Note

All F-tests are calculated via `ez::ezANOVA`, including Levene's test and Mauchly's sphericity test. (But Welch's ANOVA is calculated in case of one-way between-subject designs via `stats::oneway.test`, unless the `welch` parameter is set to `FALSE`.)

Confidence intervals are calculated, using the F value, via `MBESS::conf.limits.ncf`, coverting noncentrality parameters to partial eta squared as `ncp/(ncp+ df_nom+df_denom+1)` (Smithson, 2003).

The inclusion Bayes factor based on matched models is calculated via `bayestestR::bayesfactor_inclusion`, (with `match_models = TRUE`, and using an `BayesFactor::anovaBF` object for `models` input).

References

Girden, E. (1992). ANOVA: Repeated measures. Newbury Park, CA: Sage.

Kelley, K. (2007). Methods for the behavioral, educational, and social sciences: An R package. Behavior Research Methods, 39(4), 979-984. doi: 10.3758/BF03192993

Lakens, D. (2014). Calculating confidence intervals for Cohen's d and eta-squared using SPSS, R, and Stata [Blog post]. Retrieved from `http://daniellakens.blogspot.com/2014/06/calculating-confidence-i`
`html`

Mathot. S. (2017). Bayes like a Baws: Interpreting Bayesian Repeated Measures in JASP [Blog post]. Retrieved from `https://www.cogsci.nl/blog/interpreting-bayesian-repeated-measures-in-jasp`

McDonald, J. H. 2015. Handbook of Biological Statistics (3rd ed.). Sparky House Publishing, Baltimore, Maryland. Retrieved from http://www.biostathandbook.com

Moder, K. (2010). Alternatives to F-test in one way ANOVA in case of heterogeneity of variances (a simulation study). Psychological Test and Assessment Modeling, 52(4), 343-353.

Navarro, D. (2013). Learning Statistics with R: A Tutorial for Psychology Students and Other Beginners (Version 0.6.1). Retrieved from https://learningstatisticswithr.com/

Smithson, M. (2003). Confidence intervals. Thousand Oaks, Calif: Sage Publications.

Steiger, J. H. (2004). Beyond the F test: effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. Psychological Methods, 9(2), 164-182. doi: 10.1037/1082989X.9.2.164

See Also

t_neat

Examples

```
# assign random data in a data frame for illustration
# (note that the 'subject' is only for illustration; since each row contains the
# data of a single subject, no additional subject id is needed)
dat_1 = data.frame(
    subject = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
    grouping1 = c(1, 1, 1, 1, 2, 2, 2, 2, 2, 2),
    grouping2 = c(1, 2, 1, 2, 2, 1, 1, 1, 2, 1),
    value_1_a = c(36.2, 45.2, 41, 24.6, 30.5, 28.2, 40.9, 45.1, 31, 16.9),
    value_2_a = c(-14.1, 58.5, -25.5, 42.2, -13, 4.4, 55.5, -28.5, 25.6, -37.1),
    value_1_b = c(83, 71, 111, 70, 92, 75, 110, 111, 110, 85),
    value_2_b = c(8.024, -14.162, 3.1, -2.1, -1.5, 0.91, 11.53, 18.37, 0.3, -0.59),
    value_1_c = c(27.4,-17.6,-32.7, 0.4, 37.2, 1.7, 18.2, 8.9, 1.9, 0.4),
    value_2_c = c(7.7, -0.8, 2.2, 14.1, 22.1, -47.7, -4.8, 8.6, 6.2, 18.2)
)
head(dat_1) # see what we have

# For example, numbers '1' and '2' in the variable names of the values can
# denote sessions in an experiment, such as '_1' for first session, and '_2 for
# second session'. The letters '_a', '_b', '_c' could denote three different
# types of techniques used within each session, to be compared to each other.
# See further below for a more verbose but more meaningful example data.

# get the between-subject effect of 'grouping1'
anova_neat('dat_1', values = 'value_1_a', between_vars = 'grouping1')

# main effects of 'grouping1', 'grouping2', and their interactions
anova_neat('dat_1', values = 'value_1_a', between_vars = 'grouping1, grouping2')

# repeated measures:
# get the within-subject effect for 'value_1_a' vs. 'value_1_b'
anova_neat('dat_1', values = 'value_1_a, value_1_b')

# same, but give the factor a custom variable name
anova_neat('dat_1', values = 'value_1_a, value_1_b', within_ids = 'a_vs_b')
# or
anova_neat('dat_1', values = 'value_1_a, value_1_b', within_ids = 'letters')

# within-subject effect for 'value_1_a' vs. 'value_1_b' vs. 'value_1_c'
```

```
anova_neat('dat_1', values = 'value_1_a, value_1_b, value_1_c')

# within-subject main effect for 'value_1_a' vs. 'value_1_b' vs. 'value_1_c',
# between-subject main effect 'grouping1', and the interaction of these two main
# effects
anova_neat('dat_1', values = 'value_1_a, value_1_b, value_1_c', between_vars = 'grouping1')

# within-subject 'number' main effect for variables with number '1' vs. number
# '2' ('value_1_a' and 'value_1_b' vs. 'value_2_a' and 'value_2_b'), 'letter'
# main effect for variables with final letterr 'a' vs. final letter 'b'
# ('value_1_a' and 'value_2_a' vs. 'value_1_b' and 'value_2_b'), and the
# 'letter' x 'number' interaction
anova_neat('dat_1',
           values = 'value_1_a, value_2_a, value_1_b, value_2_b',
           within_ids = list(
               letters = c('_a', '_b'),
               numbers =  c('_1', '_2')
           ))

# same as above, but now including between-subject main effect 'grouping2' and
# its interactions
anova_neat(
    'dat_1',
    values = 'value_1_a, value_2_a, value_1_b, value_2_b',
    within_ids = list(
        letters = c('_a', '_b'),
        numbers =  c('_1', '_2')
    ),
    between_vars = 'grouping2'
)

# In real datasets, these could of course be more meaningful. For example, let's
# say participants rated the attractiveness of pictures with low or high levels
# of frightening and low or high levels of disgusting qualities. So there are
# four types of ratings:
# 'low disgusting, low frightening' pictures
# 'low disgusting, high frightening' pictures
# 'high disgusting, low frightening' pictures
# 'high disgusting, high frightening' pictures

# this could be meaningfully assigned e.g. as below
pic_ratings = data.frame(
    subject = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
   rating_fright_low_disgust_low = c(36.2, 45.2, 41, 24.6, 30.5, 28.2, 40.9, 45.1, 31, 16.9),
   rating_fright_high_disgust_low = c(-14.1, 58.5,-25.5, 42.2,-13, 4.4, 55.5,-28.5, 25.6,-37.1),
    rating_fright_low_disgust_high = c(83, 71, 111, 70, 92, 75, 110, 111, 110, 85),
   rating_fright_high_disgust_high = c(8.024,-14.162, 3.1,-2.1,-1.5, 0.91, 11.53, 18.37, 0.3,-0.59)
)
head(pic_ratings) # see what we have

# now the same logic applies as for the examples above, but now the
# within-subject differences can be more meaningfully specified, e.g.
# 'disgust_low' vs. 'disgust_high' for levels of disgustingness, while
# 'fright_low' vs. 'fright_high' for levels of frighteningness
anova_neat('pic_ratings',
           values = 'rating_fright_low_disgust_low, rating_fright_high_disgust_low,
               rating_fright_low_disgust_high, rating_fright_high_disgust_high',
            within_ids = list(
                disgustingness = c('disgust_low', 'disgust_high'),
                frighteningness =  c('fright_low', 'fright_high')
            ))
# the results are the same as for the analogous test for the 'dat_1' data, only
# with different names

# now let's say the ratings were done in two separate groups
pic_ratings = data.frame(
    subject = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
    group_id = c(1, 2, 1, 2, 2, 1, 1, 1, 2, 1),
   rating_fright_low_disgust_low = c(36.2, 45.2, 41, 24.6, 30.5, 28.2, 40.9, 45.1, 31, 16.9),
   rating_fright_high_disgust_low = c(-14.1, 58.5,-25.5, 42.2,-13, 4.4, 55.5,-28.5, 25.6,-37.1),
    rating_fright_low_disgust_high = c(83, 71, 111, 70, 92, 75, 110, 111, 110, 85),
   rating_fright_high_disgust_high = c(8.024,-14.162, 3.1,-2.1,-1.5, 0.91, 11.53, 18.37, 0.3,-0.59)
)

# now test the effect and interactions of 'group_id'
anova_neat(
    'pic_ratings',
    values = 'rating_fright_low_disgust_low, rating_fright_high_disgust_low,
        rating_fright_low_disgust_high, rating_fright_high_disgust_high',
    within_ids = list(
        disgustingness = c('disgust_low', 'disgust_high'),
        frighteningness =  c('fright_low', 'fright_high')
    ),
    between_vars = 'group_id'
)

# again, same results as with 'dat_1' (using 'grouping2' as group_id)
```

`corr_neat` *Correlation Statistics*

Description

Pearson correlation results including confidence interval (CI) and correlation Bayes factor (BF).

Usage

```
corr_neat(var1, var2, ci = 0.95, bf_added = TRUE, direction = "",
  round_r = 3, for_table = FALSE)
```

Arguments

var1	Numeric vector; numbers of the first variable.
var2	Numeric vector; numbers of the second variable.
ci	Numeric; confidence level for the returned CI, as implemented in cor.test.
bf_added	Logical. If TRUE (default), Bayes factor is calculated and displayed.

`direction`	String; optionally specifies one-sided test: either "negative" (negative correlation expected) or "positive" (positive correlation expected). (Short forms also work, e.g. "p", "pos", "neg", etc.) If left empty, the test is two-sided.
`round_r`	Number `to round` to the correlation and its CI.
`for_table`	Logical. If `TRUE`, omits the confidence level display from the printed text.

Details

The Bayes factor (BF) is always calculated with the default r-scale of `0.707`. BF supporting null hypothesis is denoted as BF01, while that supporting alternative hypothesis is denoted as BF10. When the BF is smaller than 1 (i.e., supports null hypothesis), the reciprocal is calculated (hence, BF10 = BF, but BF01 = 1/BF). When the BF is greater than or equal to 10000, scientific (exponential) form is reported for readability. (The original full BF number is available in the returned named vector as `bf`.)

Value

Prints correlation statistics (including CI and BF) in APA style. Furthermore, when assigned, returns a named vector with the following elements: `r` (Pearson correlation), `p` (p value), `bf` (Bayes factor).

Note

The correlation and CI is calculated via `stats::cor.test`.

The Bayes factor is calculated via `BayesFactor::correlationBF`.

See Also

`t_neat`

Examples

```
# assign two variables
v1 = c(11, 15, 19, 43, 53, -4, 34, 8, 33, -1, 54 )
v2 = c(4, -2, 23, 13, 32, 16, 3, 29, 37, -4, 65 )

corr_neat(v1, v2) # prints statistics

# one-sided, and omitting the "95% CI" part
corr_neat(v1, v2, direction = 'pos', for_table = TRUE)

# print statistics and assign main results
results = corr_neat(v1, v2, direction = 'pos')

results['p'] # get precise p value
```

dems_neat *Demographics*

Description

Prints participant count, age mean and SD, and gender ratio, from given dataset.

Usage

```
dems_neat(data_per_subject, group_by = NULL, percent = TRUE,
  round_perc = 0)
```

Arguments

`data_per_subject`	Data frame from which demographics are to be calculated. Must contain columns named precisely as "age" and as "gender". The "age" column must contain numbers, while "gender" column must contain 1 (= male) and 2 (= female) only (either as numbers or as strings).
`group_by`	A vector of factors by which the statistics are grouped, typically a column from the data frame provided as `data_per_subject`.
`percent`	Logical. If `FALSE`, gender ratio is presented as count of males. Otherwise (default), presented as percent of males.
`round_perc`	Number `to round` to, when using percents.

Examples

```
# below is an illustrative example dataset
# (the "subject" and "measure_x" columns are not used in the function)
dat = data.frame(
    subject = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
    conditions = c('x', 'y', 'x', 'y', 'y', 'x', 'x', 'x', 'y', 'x'),
    gender = c(2, 2, 1, 2, 1, 2, 2, 2, 1, 1),
    age = c(6, 7, 8.5, 6, 5, 16, 17, 16, 45, 77),
    measure_x = c(83, 71, 111, 70, 92, 75, 110, 111, 110, 85)
)

# print demographics (age and gender) per "conditions":
dems_neat(dat, group_by = dat$conditions)
```

mon_conv *Monitor Screen Unit Conversion*

Description

Given a specific monitor object, converts specified screen units to other specified units. The possible units to convert from and to: "cm" (centimeters), "pix" (pixels), or "deg" (degrees of visual angle).

Usage

```
mon_conv(mon_obj, value, from, to)
```

Arguments

`mon_obj`	Monitor object, as assigned with `mon_neat`.
`value`	Number; value (magnitude) of the given unit to convert from.
`from`	String; unit ("cm", "pix", or "deg") to convert from.
`to`	String; unit ("cm", "pix", or "deg") to convert to.

Value

Number (magnitude) in the given output (`to`) unit.

See Also

`mon_neat`, `mon_params`

Examples

```
# assign monitor with 50 cm distance, screen width 52 cm and 1920 pixels
my_mon = mon_neat(distance = 50, mon_width_cm = 52, mon_width_pixel = 1920)

# convert 30.4 pixels to degrees of visual angle, for the specified monitor
mon_conv(my_mon, 30.4, 'pix', 'deg') # returns 0.9434492 (degrees)

# convert 0.94 degrees of visual angle to pixels
mon_conv(my_mon, 0.94, 'deg', 'pix') # returns 30.28885 (pixels)

# convert 10 degrees of visual angle to cm
mon_conv(my_mon, 10, from = 'deg', to = 'cm')

# convert 8.748866 cm to pixels
mon_conv(my_mon, 8.748866, from = 'cm', to = 'pix')
```

`mon_neat` *Monitor Object*

Description

Assigns a monitor object, storing distance and width parameters.

Usage

```
mon_neat(distance, mon_width_cm, mon_width_pixel)
```

Arguments

`distance`	Viewing distance in cm (from eyes to screen).
`mon_width_cm`	Monitor screen width in cm.
`mon_width_pixel`	Monitor screen width in pixels.

Value

A monitor object with the specified parameters, to be used in the `mon_conv` function.

See Also

mon_conv, mon_params

Examples

```
# assign monitor with 57 cm viewing distance, screen width 52 cm and 1920 pixels
my_mon = mon_neat(distance = 57, mon_width_cm = 52, mon_width_pixel = 1920)
```

---

mon_params *Monitor parameters*

Description

Prints the parameters of a given monitor object.

Usage

```
mon_params(mon_obj)
```

Arguments

mon_obj	Monitor object, as assigned with mon_neat.

Value

Returns nothing, just prints all parameters.

See Also

mon_neat, mon_conv

Examples

```
# assign monitor with 57 cm distance, and screen width 52 cm and 1920 pixels
my_mon = mon_neat(distance = 57, mon_width_cm = 52, mon_width_pixel = 1920)

# print the above given parameters
mon_params( my_mon )
```

m_neat *Neat Descriptives*

Description

Returns means (or medians) and SDs per group for given variable. Primarily for use in the `table_neat` function.

Usage

```
m_neat(values, round_to = 0, new_name = NULL, group_by = NULL,
  medians = FALSE)
```

Arguments

`values`	A vector of numbers from which the statistics are to be calculated.
`round_to`	Number of digits after the decimal point to round to.
`new_name`	String. A new name for the variable to be used as column title.
`group_by`	A vector of factors by which the statistics are grouped.
`medians`	Logical. If `TRUE`, medians are calculated, otherwise means.

Value

A data frame with the statistics per group. Furthermore, prints statistics per group, unless used within the `table_neat` function, in which case nothing is printed.

See Also

`table_neat` to create full tables using multiple variables

Examples

```
data("mtcars") # load base R example dataset

res1 = m_neat(mtcars$wt ) # overall means and SDs for wt (Weight)

res2 = m_neat(mtcars$wt, 2, new_name = 'weight') # rounded to 2 and renamed

res3 = m_neat(mtcars$wt, 2, group_by = mtcars$cyl) # grouped by cyl (Number of cylinders)

res4 = m_neat(mtcars$wt, 2, group_by = mtcars$cyl, medians = TRUE) # medians
```

props_neat *Difference of Two Proportions*

Description

Unconditional exact test results for the comparison of two independent proportions, including confidence interval (CI) for the proportion difference, and corresponding independent multinomial contingency table Bayes factor (BF). Cohen's h and its CI are also calculated.

Usage

```
props_neat(case1, case2, n1, n2, greater = "", ci = NULL,
  bf_added = TRUE, h_added = FALSE, for_table = FALSE)
```

Arguments

`case1`	Number of 'cases' (as opposed to 'controls'; e.g. positive outcomes vs. negative outcomes) in 'group 1'.
`case2`	Number of 'cases' in 'group 2'.
`n1`	Number; sample size of 'group 1'.
`n2`	Number; sample size of 'group 2'.
`greater`	String (or number); optionally specifies one-sided exact test: either "1" (`case1/n1` proportion expected to be greater than `case2/n2` proportion) or "2" (`case2/n2` proportion expected to be greater than `case1/n1` proportion). If left empty, the test is two-sided.
`ci`	Numeric; confidence level for the returned CIs (proportion difference and Cohen's h).
`bf_added`	Logical. If `TRUE` (default), Bayes factor is calculated and displayed. (Always two-sided.)
`h_added`	Logical. If `TRUE`, Cohen's h and its CI are calculated and displayed. (`FALSE` by default.)
`for_table`	Logical. If `TRUE`, omits the confidence level display from the printed text.

Details

The Bayes factor (BF) is always calculated with the default r-scale of `0.707`. BF supporting null hypothesis is denoted as BF01, while that supporting alternative hypothesis is denoted as BF10. When the BF is smaller than 1 (i.e., supports null hypothesis), the reciprocal is calculated (hence, BF10 = BF, but BF01 = 1/BF). When the BF is greater than or equal to 10000, scientific (exponential) form is reported for readability. (The original full BF number is available in the returned named vector as `bf`.)

Value

Prints exact test statistics (including proportion difference with CI, and BF) in APA style. Furthermore, when assigned, returns a named vector with the following elements: `z` (Z), `p` (p value), `prop_diff` (raw proportion difference), `h` (Cohen's h), `bf` (Bayes factor).

Note

Barnard's unconditional exact test is calculated via `Exact::exact.test` ("z-pooled").

The CIs for the proportion difference is calculated based on the p value, as described by Altman and Bland (2011).

The Bayes factor is calculated via `BayesFactor::contingencyTableBF`, with `sampleType = "indepMulti"`, as appropriate when both sample sizes (`n1` and `n2`) are known in advance (as it normally happens). (For details, see `contingencyTableBF`, or e.g. 'Chapter 17 Bayesian statistics' in Navarro, 2019.)

References

Altman, D. G., & Bland, J. M. (2011). How to obtain the confidence interval from a P value. Bmj, 343(d2090). doi: 10.1136/bmj.d2090

Barnard, G. A. (1947). Significance tests for 2x2 tables. Biometrika, 34(1/2), 123-138. doi: 10.1093/biomet/34.12.123

Lydersen, S., Fagerland, M. W., & Laake, P. (2009). Recommended tests for association in 2x2 tables. Statistics in medicine, 28(7), 1159-1175. doi: 10.1002/sim.3531

Navarro, D. (2019). Learning statistics with R. `https://learningstatisticswithr.com/`

Suissa, S., & Shuster, J. J. (1985). Exact unconditional sample sizes for the 2 times 2 binomial trial. Journal of the Royal Statistical Society: Series A (General), 148(4), 317-327. doi: 10.2307/2981892

Examples

```
props_neat(
    case1 = 35,
    case2 = 48,
    n1 = 80,
    n2 = 77,
    h_added = TRUE
)

props_neat(
    case1 = 35,
    case2 = 48,
    n1 = 80,
    n2 = 77,
    greater = "2"
)
```

`ro` *Neat rounding*

Description

Rounds a given number to given number of digits after the decimal point, returning it as string, with trailing zeros when applicable.

Usage

```
ro(num, round_to = 2)
```

Arguments

`num`	Number to be rounded.
`round_to`	Number of fractional digits (i.e., digits after the decimal point), to round to.

Value

Number as string: `num` rounded to `round_to` digits, with trailing zeros when applicable.

Examples

```
ro( 1.2345 ) # returns "1.23"

ro( 0.12345, 1 ) # returns "0.1"

ro( 12.3, 4 ) # returns "12.3000"
```

`roc_neat` *Difference of Two Areas Under the Curves*

Description

Comparison of two `areas under the receiver operating characteristic curves` (AUCs).

Usage

```
roc_neat(roc1, roc2, pair = FALSE, greater = "")
```

Arguments

`roc1`	Receiver operating characteristic (ROC) `object`.
`roc2`	Receiver operating characteristic (ROC) `object`.
`pair`	Logical. If `TRUE`, the test is conducted for paired samples. Otherwise (default) for independent samples.
`greater`	String (or number); optionally specifies one-sided test: either "1" (`roc1` AUC expected to be greater than `roc2` AUC) or "2" (`roc2` AUC expected to be greater than `roc2` AUC). If left empty, the test is two-sided.

Value

Prints DeLong's test results for the comparison of the two given AUCs in APA style. Furthermore, when assigned, returns a named vector with the following two elements: `stat` (D value), `p` (p value).

Note

The test statistics are calculated via `pROC::roc.test` as DeLong's test (for both paired and unpaired). The `roc_neat` function merely prints it in APA style.

The ROC object may be calculated via `t_neat`, or directly with `pROC::roc`.

References

DeLong, E. R., DeLong, D. M., & Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. Biometrics, 44(3), 837-845. doi: 10.2307/2531595

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J. C., & Muller, M. (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC bioinformatics, 12(1), 77. doi: 10.1186/147121051277

See Also

`t_neat`

Examples

```
# calculate first AUC (from v1 and v2)
v1 = c(191, 115, 129, 43, 523,-4, 34, 28, 33,-1, 54)
v2 = c(4,-2, 23, 13, 32, 16, 3, 29, 37,-4, 65)
results1 = t_neat(v1, v2, auc_added = TRUE)

# calculate second AUC (from v3 and v4)
v3 = c(14.1, 58.5, 25.5, 42.2, 13, 4.4, 55.5, 28.5, 25.6, 37.1)
v4 = c(36.2, 45.2, 41, 24.6, 30.5, 28.2, 40.9, 45.1, 31, 16.9)
results2 = t_neat(v3, v4, auc_added = TRUE)

# one-sided comparison of the two AUCs
roc_neat(results1$roc_obj, results2$roc_obj, greater = "1")
```

`script_path` *Script Path*

Description

Gives, in RStudio, the path to the script file in which it is executed.

Usage

```
script_path(subdir = "")
```

Arguments

`subdir`	String, optional. Subdirectory relative to the script's path.

Value

Script file's path as string. If `subdir` is given, it is appended to the original path.

Examples

```
setwd( script_path() ) # sets working directory to the script's path, e.g. "C:/script_folder/"


script_path('my_subdir/misc/') # returns "C:/script_folder/my_subdir/misc/"
```

table_neat *Neat Table*

Description

Creates a neat means (or medians) and standard deviations table, using m_neat() function(s) as arguments.

Usage

```
table_neat(values_list, group_by = NULL, group_per = "rows",
  to_clipboard = FALSE, medians = NULL)
```

Arguments

values_list	Values returned (as data frames) from the m_neat function: variables from which the statistics for the table are to be calculated. The group_by and medians parameters are ignored when they are given in the table_neat function; see Details.
group_by	A vector of factors by which the statistics are grouped. (Overwrites group_by in m_neat; see Details.)
group_per	String, "rows" or "columns". If set to "columns" (or just "c" or "col", etc.), each column contains statistics for one group. Otherwise (default), each row contains statistics for one group.
to_clipboard	Logical. If TRUE, the table is copied to the clipboard.
medians	Logical. If TRUE, medians are calculated, otherwise means. (Overwrites medians in m_neat; see Details.)

Details

The values, round_to, and new_name arguments given in the m_neat function are always applied. However, the group_by or medians given in the m_neat function are only applied when no arguments are given in the table_neat function for the identical parameters (group_by or medians). If either parameter is given in the table_neat function, all separately given respective argument(s) in the m_neat function(s) are ignored.

Value

Returns a data frame with means or medians and SDs per variable and per group.

See Also

m_neat for more related details

Examples

```
data("mtcars") # load base R example dataset

# overall means and SDs table for disp (Displacement) and hp (Gross horsepower)
Ms_SDs = table_neat(list(m_neat(mtcars$disp),
                         m_neat(mtcars$hp)))
```

```
# means and SDs table for mpg (Miles/(US) gallon), wt (Weight), and hp (Gross horsepower)
# grouped by cyl (Number of cylinders)
# each measure rounded to respective optimal number of digits
# wt renamed to weight (for the column title)
Ms_SDs2 = table_neat(list(m_neat(mtcars$mpg, 1),
                          m_neat(mtcars$wt, 2, new_name = 'weight'),
                          m_neat(mtcars$hp)),
                     group_by = mtcars$cyl)

# same as above, but with medians, and with groups per columns
Ms_SDs3 = table_neat(list(m_neat(mtcars$mpg, 1),
                          m_neat(mtcars$wt, 2, new_name = 'weight'),
                          m_neat(mtcars$hp)),
                     group_by = mtcars$cyl,
                     medians = TRUE,
                     group_per = 'columns')
```

`t_neat` *Difference of Two Means and Area Under the Curve*

Description

Welch's `t-test` results including Cohen's d with confidence interval (CI), `Bayes factor` (BF), and `area under the receiver operating characteristic curve` (AUC).

Usage

```
t_neat(var1, var2, pair = FALSE, greater = "", ci = NULL,
  bf_added = TRUE, auc_added = FALSE, r_added = TRUE,
  for_table = FALSE, test_title = "Descriptives:", round_descr = 2,
  round_auc = 3, auc_greater = "")
```

Arguments

`var1`	Numeric vector; numbers of the first variable.
`var2`	Numeric vector; numbers of the second variable.
`pair`	Logical. If `TRUE`, all tests (t, BF, AUC) are conducted for paired samples. Otherwise (default) for independent samples.
`greater`	String (or number); optionally specifies one-sided tests (t and BF): either "1" (`var1` mean expected to be greater than `var2` mean) or "2" (`var2` mean expected to be greater than `var1` mean). If left empty, the test is two-sided.
`ci`	Numeric; confidence level for returned CIs for Cohen's d and AUC.
`bf_added`	Logical. If `TRUE` (default), Bayes factor is calculated and displayed.
`auc_added`	Logical. If `TRUE`, AUC is calculated and displayed. (`FALSE` by default.)
`r_added`	Logical. If `TRUE` (default), Pearson correlation is calculated and displayed in case of paired comparison.
`for_table`	Logical. If `TRUE`, omits the confidence level display from the printed text.

`test_title`	String, "Descriptives:" by default. Simply displayed in printing preceding the descriptive statistics. (Useful e.g. to distinguish several different comparisons inside a `function` or a `for` loop.)
`round_descr`	Number `to round` to the descriptive statistics (means and SDs).
`round_auc`	Number `to round` to the AUC and its CI.
`auc_greater`	String (or number); specifies which variable is expected to have greater values for 'cases' as opposed to 'controls': "1" (default; `var1` expected to be greater for 'cases' than `var2` mean) or "2" (`var2` expected to be greater for 'cases' than `var1`). Not to be confused with one-sided tests; see Details.

Details

The Bayes factor (BF) is always calculated with the default r-scale of `0.707`. BF supporting null hypothesis is denoted as BF01, while that supporting alternative hypothesis is denoted as BF10. When the BF is smaller than 1 (i.e., supports null hypothesis), the reciprocal is calculated (hence, BF10 = BF, but BF01 = 1/BF). When the BF is greater than or equal to 10000, scientific (exponential) form is reported for readability. (The original full BF number is available in the returned named vector as `bf`.)

The original `pROC::auc` function, by default, always returns an AUC greater than (or equal to) .5, assuming that the prediction based on values in the expected direction work correctly at least at chance level. This however may be confusing. Consider an example where we measure the heights of persons in a specific small sample and expect that greater height predicts masculine gender. The results are, say, 169, 175, 167, 164 (cm) for one gender, and 176, 182, 179, 165 for the other. If the expectation is correct (the second, greater values are for males), the AUC is .812. However, if in this particular population females are actually taller than males, the AUC is in fact .188. To keep things clear, the `t_neat` function always makes an assumption about which variable is expected to be greater for correct classification ("1" by default; i.e., `var1`; to be specified as `auc_greater = "2"` for `var2` to be expected as greater). For this example, if the first (smaller) variables are given as `var1` for females, and second (larger), variables are given as `var2` for males, we have to specify `auc_greater = "2"` to indicate the expectation of larger values for males. (Or, easier, just add the expected larger values as `var1`.)

Value

Prints t-test statistics (including Cohen'd with CI, BF, and AUC, as specified via the corresponding parameters) in APA style. Furthermore, when assigned, returns a list, that contains a named vector '`stats`' with the following elements: `t` (t value), `p` (p value), `d` (Cohen's d), `bf` (Bayes factor), `auc` (AUC), `accuracy` (accuracy using the most optimal classification threshold). The latter two is `NULL` when `auc_added` is `FALSE`. When `auc_added` is `TRUE`, there are also two additional elements of the list. One is '`roc_obj`', which is a `roc` object, to be used e.g. with the `roc_neat` function. The other is '`best_thresholds`', which contains the best threshold value(s) for classification, along with corresponding specificity and sensitivity.

Note

The Welch's t-test is calculated via `stats::t.test`.

Cohen's d and its confidence interval are calculated, using the t value, via `MBESS::ci.smd` for independent samples (as standardized mean difference) and via `MBESS::ci.sm` for paired samples (as standardized mean).

The Bayes factor is calculated via `BayesFactor::ttestBF`.

The correlation and its CI are calculated via `stats::cor.test`, and is always two-sided, always with 95 percent CI. For more, use `corr_neat`.

The AUC and its CI are calculated via `pROC::auc`, and the accuracy at most optimal threshold via `pROC::coords` (`x = "best"`); both using the object `pROC::roc`.

References

Delacre, M., Lakens, D., & Leys, C. (2017). Why psychologists should by default use Welch's t-test instead of Student's t-test. International Review of Social Psychology, 30(1). doi: 10.5334/irsp.82

Kelley, K. (2007). Methods for the behavioral, educational, and social sciences: An R package. Behavior Research Methods, 39(4), 979-984. doi: 10.3758/BF03192993

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J. C., & Muller, M. (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC bioinformatics, 12(1), 77. doi: 10.1186/147121051277

See Also

`corr_neat`, `roc_neat`

Examples

```
# assign two variables (numeric vectors)
v1 = c(191, 115, 129, 43, 523,-4, 34, 28, 33,-1, 54)
v2 = c(4,-2, 23, 13, 32, 16, 3, 29, 37,-4, 65)

t_neat(v1, v2) # prints results as independent samples
t_neat(v1, v2, pair = TRUE) # as paired samples (r added by default)
t_neat(v1, v2, pair = TRUE, greater = "1") # one-sided
t_neat(v1, v2, pair = TRUE, auc_added = TRUE ) # AUC included

# print results and assign returned list
results = t_neat(v1, v2, pair = TRUE)

results$stats['bf'] # get precise BF value
```

Index