Package ‘bruceR’

April 13, 2022

Title Broadly Useful Convenient and Efficient R Functions

Version 0.8.6

Date 2022-04-13

Author Han-Wu-Shuang Bao [aut, cre]

Maintainer Han-Wu-Shuang Bao <baohws@foxmail.com>

Description Broadly useful convenient and efficient R functions that bring users concise and elegant R data analyses. This package includes easy-to-use functions for

1) basic R programming
   (e.g., set working directory to the path of currently opened file, import/export data from/to files with any format, print strings with rich formats and colors);
2) multivariate computation
   (e.g., compute scale sums/means/... with reverse scoring);
3) reliability analyses and factor analyses;
4) descriptive statistics and correlation analyses;
5) t-test, multi-factor analysis of variance (ANOVA), simple-effect analysis, and post-hoc multiple comparison;
6) tidy report of statistical models
   (to R Console and Microsoft Word);
7) mediation and moderation analyses (PROCESS);
and (8) additional toolbox for statistics and graphics.

License GPL-3

Encoding UTF-8

LazyData true

URL https://psychbruce.github.io/bruceR/

BugReports https://github.com/psychbruce/bruceR/issues

Depends R (>= 4.0.0)

Imports dplyr, tidyr, stringr, forcats, data.table, psych, afex,
   emmeans, effectsize, performance, lmerTest, mediation,
   interactions, lavaan, glue, crayon, ggplot2, ggtext, cowplot,
   see

1
Suggests rstudioapi, pacman, rio, haven, foreign, readxl, openxlsx, clipr, tibble, plyr, car, phia, lmtest, lme4, vars, GPArotation, jtools, texreg, MuMIn, BayesFactor, GGally

RoxygenNote 7.1.2
NeedsCompilation no
Repository CRAN
Date/Publication 2022-04-13 15:02:29 UTC

R topics documented:

bruceR-package ........................................... 3
Alpha ................................................... 5
bruceR-demodata ........................................ 6
cr ........................................................... 7
ccf_plot .................................................. 8
CFA ....................................................... 9
Corr ....................................................... 11
cor_diff .................................................. 12
Describe .................................................. 13
dtime ..................................................... 15
EFA .......................................................... 15
EMMEANS ................................................ 18
export .................................................... 23
formatF ................................................... 25
formatN ................................................... 25
formula_expand ........................................... 26
formula_paste .......................................... 27
Freq ........................................................ 27
GLM_summary ............................................ 28
grand_mean_center ..................................... 29
granger_causality ...................................... 30
granger_test ............................................ 31
group_mean_center ..................................... 32
HLM_ICC_rWG ........................................... 33
HLM_summary ............................................ 35
import .................................................... 36
lavaan_summary ......................................... 38
LOOKUP .................................................. 40
MANOVA .................................................. 42
med_summary ........................................... 46
model_summary ......................................... 47
p ............................................................... 49
pkg_depend ............................................. 51
pkg_install_suggested ................................ 51
Print ..................................................... 52
print_table ............................................. 53
Description

BRoadly Useful Convenient and Efficient R functions that BRing Users Concise and Elegant R data analyses.

Package homepage: https://psychbruce.github.io/bruceR/

Install the latest development version from GitHub: devtools::install_github("psychbruce/bruceR")

Report bugs at GitHub Issues.

Details

Loading bruceR by library(bruceR) will also load these R packages for you:

[Data]:

- dplyr: Data manipulation and processing.
- tidyr: Data cleaning and reshaping.
- stringr: Toolbox for string operation (with regular expressions).
- forcats: Toolbox for factor manipulation (for categorical variables).
- data.table: Advanced data.frame with higher efficiency.

[Stat]:
• emmeans: Estimates of marginal means and multiple contrasts.
• effectsize: Estimates of effect sizes and standardized parameters.
• lmerTest: Tests of linear mixed effects models (LMM, also known as HLM and multilevel models).

[Plot]:
• ggplot2: Data visualization.
• ggtext: Markdown/HTML rich text format for ggplot2 (geoms and themes).
• cowplot: Advanced toolbox for ggplot2 (arrange multiple plots and add labels).
• see: Advanced toolbox for ggplot2 (geoms, scales, themes, and color palettes).

Main Functions in bruceR

(1) Basic R Programming set.wd (alias: set wd)
import, export
cc
pkg_depend, pkg_install_suggested
formatF, formatN
print_table
Print, Glue, Run
%^% %notin% %allin%, %anyin%, %nonein%, %partin%

(2) Multivariate Computation SUM, MEAN, STD, MODE, COUNT, CONSEC
RECODE, RESCALE
LOOKUP

(3) Reliability and Factor Analyses Alpha
EFA / PCA
CFA

(4) Descriptive Statistics and Correlation Analyses Describe
Freq
Corr
cor_diff

(5) T-Test, Multi-Factor ANOVA, Simple-Effect Analysis, and Post-Hoc Multiple Comparison
TTEST
MANOVA
EMMEANS

(6) Tidy Report of Regression Models model_summary
lavaan_summary
GLM_summary
HLM_summary
HLM_ICC_rWG
regress
Note

Please always use RStudio as an IDE instead of using the raw R software.

Author(s)

Han-Wu-Shuang (Bruce) Bao
Email: <baohws@foxmail.com>

Description

An extension of \texttt{psych::alpha()} and \texttt{psych::omega()}, reporting (1) scale statistics (Cronbach’s $\alpha$ and McDonald’s $\omega$) and (2) item statistics (item-rest correlation [i.e., corrected item-total correlation] and Cronbach’s $\alpha$ if item deleted).

Three options to specify variables:

1. \texttt{var + items}: use the common and unique parts of variable names.
2. \texttt{vars}: directly define a character vector of variables.
3. \texttt{varrange}: use the starting and stopping positions of variables.

Usage

\begin{verbatim}
Alpha(
data,
        var,
        items,
        vars = NULL,
        varrange = NULL,
        rev = NULL,
        digits = 3,
        nsmall = digits
    )
\end{verbatim}
Arguments

data  Data frame.
var   [Option 1] The common part across the variables. e.g., "RSES"
items  [Option 1] The unique part across the variables. e.g., 1:10
vars  [Option 2] A character vector specifying the variables. e.g., c("X1", "X2", "X3", "X4", "X5")
varrange  [Option 3] A character string specifying the positions ("starting:stopping") of variables. e.g., "A1:E5"
rev  [Optional] Variables that need to be reversed. It can be (1) a character vector specifying the reverse-scoring variables (recommended), or (2) a numeric vector specifying the item number of reverse-scoring variables (not recommended).
digits, nsmall  Number of decimal places of output. Default is 3.

Value

A list of results obtained from `psych::alpha()` and `psych::omega()`.

See Also

`MEAN`, `EFA`, `CFA`

Examples

```r
# ?psych::bfi
data = psych::bfi
Alpha(data, "E", 1:5)  # "E1" & "E2" should be reversed
Alpha(data, "E", 1:5, rev=1:2)  # correct
Alpha(data, "E", 1:5, rev=c("E1", "E2"))  # also correct
Alpha(data, vars=c("E1", "E2", "E3", "E4", "E5"), rev=c("E1", "E2"))
Alpha(data, varrange="E1:E5", rev=c("E1", "E2"))

# using dplyr::select()
data %>% select(E1, E2, E3, E4, E5) %>%
  Alpha(vars=names(.), rev=c("E1", "E2"))
```

Description

Demo datasets of multi-factor ANOVA as examples to show how the functions `MANOVA` and `EMMEANS` work.
Format

1. Between-Subjects Design
   • between.1 - A(4)
   • between.2 - A(2) * B(3)
   • between.3 - A(2) * B(2) * C(2)

2. Within-Subjects Design
   • within.1 - A(4)
   • within.2 - A(2) * B(3)
   • within.3 - A(2) * B(2) * C(2)

3. Mixed Design
   • mixed.2_1b1w - A(2, between) * B(3, within)
   • mixed.3_1b2w - A(2, between) * B(2, within) * C(2, within)
   • mixed.3_2b1w - A(2, between) * B(2, within) * C(2, between)

Source

Multi-Factor Experimental Design in Psychology and Education

---

cc

Split up a string (with separators) into a character vector.

Description

Split up a string (with separators) into a character vector (whitespace around separator is trimmed).

Usage

cc(..., sep = "auto")

Arguments

... Character string(s).
sep Pattern for separation.
   Default is "auto", including 5 common separators: , ; | \n \t.

Value

Character vector.

Examples

cc("a,b,c,d,e")
cc(" a , b , c , d , e ")
cc("1, 2, 3, 4, 5")
cc("A 1 , B 2 ; C 3 | D 4 \t E 5")
ccf_plot

Cross-correlation analysis.

Description
Plot the results of cross-correlation analysis using ggplot2 (rather than R base plot) for more flexible modification of the plot.

Usage
ccf_plot(
  formula,
  data,
  lag.max = 30,
  sig.level = 0.05,
  xbreaks = seq(-100, 100, 10),
  ybreaks = seq(-1, 1, 0.2),
  ylim = NULL,
  alpha.ns = 1,
  pos.color = "black",
  neg.color = "black",
  ci.color = "blue",
  title = NULL,
  subtitle = NULL,
  xlab = "Lag",
  ylab = "Cross-Correlation"
)

Arguments

formula | Model formula like y ~ x.
data | Data frame.
lag.max | Maximum time lag. Default is 30.
sig.level | Significance level. Default is 0.05.
xbreaks | X-axis breaks.
CFA

ybreaks  Y-axis breaks.
ylim     Y-axis limits. Default is NULL to automatically estimate.
alpha.ns Color transparency (opacity: 0~1) for non-significant values. Default is 1 for no transparency (i.e., opaque color).
pos.color Color for positive values. Default is "black".
neg.color Color for negative values. Default is "black".
ci.color Color for upper and lower bounds of significant values. Default is "blue".
title    Plot title. Default is an illustration of the formula.
subtitle Plot subtitle.
xlab     X-axis title. Default is "Lag".
ylab     Y-axis title. Default is "Cross-Correlation".

Details

Significant correlations with negative time lags suggest shifts in a predictor precede shifts in an outcome.

Value

A gg object, which you can further modify using ggplot2 syntax and save using ggsave().

See Also

ganger_test

Examples

# resemble the default plot output by `ccf()`
p1 = ccf_plot(chicken ~ egg, data=lmtest::ChickEgg)

# a more colorful plot
p2 = ccf_plot(chicken ~ egg, data=lmtest::ChickEgg, alpha.ns=0.3,
pos.color="#CD201F",
neg.color="#21759B",
ci.color="black")

CFA

Confirmatory Factor Analysis (CFA).

Description

An extension of lavaan::cfa().
Usage

CFA(
  data,
  model = "A =~ a[1:5]; B =~ b[c(1,3,5)]; C =~ c1 + c2 + c3",
  estimator = "ML",
  highorder = "",
  orthogonal = FALSE,
  missing = "listwise",
  digits = 3,
  nsmall = digits,
  file = NULL
)

Arguments

data          Data frame.
model         Model formula. See examples.
estimator     The estimator to be used (for details, see lavaan options). Default is "ML". Can be one of the following:
"ML"  Maximum Likelihood (can be extended to "MLM", "MLMV", "MLMVS", "MLF", or "MLR" for robust standard errors and robust test statistics)
"GLS"  Generalized Least Squares
"WLS"  Weighted Least Squares
"ULS"  Unweighted Least Squares
"DWLS"  Diagonally Weighted Least Squares
"DLS"  Distributionally-weighted Least Squares
highorder     High-order factor. Default is "".
orthogonal    Default is FALSE. If TRUE, all covariances among latent variables are set to zero.
missing       Default is "listwise". Alternative is "fiml" ("Full Information Maximum Likelihood").
digits, nsmall Number of decimal places of output. Default is 3.
file          File name of MS Word (.doc).

Value

A list of results returned by lavaan::cfa().

See Also

Alpha, EFA, lavaan_summary

Examples

data.cfa=lavaan::HolzingerSwineford1939
CFA(data.cfa, "Visual =~ x[1:3]; Textual =~ x[c(4,5,6)]; Speed =~ x7 + x8 + x9")
CFA(data.cfa, model="
Corr

Visual =~ x[1:3]
Textual =~ x[c(4,5,6)]
Speed =~ x7 + x8 + x9
"", highorder="Ability")

data.bfi = na.omit(psych::bfi)
CFA(data.bfi, "E =~ E[1:5]; A =~ A[1:5]; C =~ C[1:5]; N =~ N[1:5]; O =~ O[1:5]"

Correlation analysis.

Description

Correlation analysis.

Usage

Corr(
  data,
  method = "pearson",
  p.adjust = "none",
  all.as.numeric = TRUE,
  digits = 2,
  nsmall = digits,
  file = NULL,
  plot = TRUE,
  plot.range = c(-1, 1),
  plot.palette = NULL,
  plot.color.levels = 201,
  plot.file = NULL,
  plot.width = 8,
  plot.height = 6,
  plot.dpi = 500
)

Arguments

data          Data frame.
method         "pearson" (default), "spearman", or "kendall".
p.adjust       Adjustment of p values for multiple tests: "none", "fdr", "holm", "bonferroni",
                ... For details, see stats::p.adjust().
all.as.numeric TRUE (default) or FALSE. Transform all variables into numeric (continuous).
digits, nsmall Number of decimal places of output. Default is 2.
file           File name of MS Word (.doc).
plot           TRUE (default) or FALSE. Plot the correlation matrix.
cor_diff

Description
Test the difference between two correlations.

Usage
cor_diff(r1, n1, r2, n2, n = NULL, rcov = NULL)
Describe

Arguments

- **r1, r2** Correlation coefficients (Pearson’s r).
- **n, n1, n2** Sample sizes.
- **rcov** [optional] Only for nonindependent rs: r1 is r(X,Y), r2 is r(X,Z), then, as Y and Z are also correlated, we should also consider rcov: r(Y,Z)

Value

Invisibly return the p value.

Examples

- # two independent rs (X~Y vs. Z~W)
  cor_diff(r1=0.20, n1=100, r2=0.45, n2=100)

- # two nonindependent rs (X~Y vs. X~Z, with Y and Z also correlated [rcov])
  cor_diff(r1=0.20, r2=0.45, n=100, rcov=0.80)

Describe **Descriptive statistics.**

Description

Descriptive statistics.

Usage

```
Describe(
  data,
  all.as.numeric = TRUE,
  digits = 2,
  nsmall = digits,
  file = NULL,
  plot = FALSE,
  upper.triangle = FALSE,
  upper.smooth = "none",
  plot.file = NULL,
  plot.width = 8,
  plot.height = 6,
  plot.dpi = 500
)
```
Describe

Arguments

data Data frame or numeric vector.
all.as.numeric TRUE (default) or FALSE. Transform all variables into numeric (continuous).
digits, nsmall Number of decimal places of output. Default is 2.
file File name of MS Word (.doc).
plot TRUE or FALSE (default). Visualize the descriptive statistics using \texttt{GGally::ggpairs()}
upper.triangle TRUE or FALSE (default). Add (scatter) plots to upper triangle (time consuming when sample size is large).
upper.smooth "none" (default), "lm", or \texttt{loess"}. Add fitting lines to scatter plots (if any).
plot.file NULL (default, plot in RStudio) or a file name ("xxx.png").
plot.width Width in "inch" of the saved plot. Default is 8.
plot.height Height in "inch" of the saved plot. Default is 6.
plot.dpi DPI (dots per inch) of the saved plot. Default is 500.

Value

Invisibly return a list consisting of (1) a data frame of descriptive statistics and (2) a \texttt{ggplot2} object if users set plot=TRUE.

See Also

\texttt{Corr}

Examples

\texttt{set.seed(1)}
\texttt{Describe(rnorm(1000000), plot=TRUE)}
\texttt{Describe(airquality)}
\texttt{Describe(airquality, plot=TRUE, upper.triangle=TRUE, upper.smooth="lm")}
\texttt{# \texttt{psych::bfi}}
\texttt{Describe(psych::bfi[c("age", "gender", "education")])}
\texttt{d = as.data.table(psych::bfi)}
\texttt{d[, \textquote{age, gender, education}]:=}
\texttt{(}
\texttt{  gender = as.factor(gender),}
\texttt{  education = as.factor(education),}
\texttt{  E = MEAN(d, \textquote{E}, 1:5, rev=c(1,2), likert=1:6),}
\texttt{  A = MEAN(d, \textquote{A}, 1:5, rev=1, likert=1:6),}
\texttt{  C = MEAN(d, \textquote{C}, 1:5, rev=c(4,5), likert=1:6),}
\texttt{  N = MEAN(d, \textquote{N}, 1:5, likert=1:6),}
\texttt{  O = MEAN(d, \textquote{O}, 1:5, rev=c(2,5), likert=1:6)}
\texttt{)]}
\texttt{Describe(d[, \textquote{age, gender, education}]), plot=TRUE, all.as.numeric=FALSE)}
\texttt{Describe(d[, \textquote{age, gender, education, E, A, C, N, O}]), plot=TRUE) }
dt ime

Timer (compute time difference).

Description

Timer (compute time difference).

Usage

dtime(t0, unit = "secs", digits = 0, nsmall = digits)

Arguments

t0 Time at the beginning.
unit Options: "auto", "secs", "mins", "hours", "days", "weeks". Default is "secs".
digits, nsmall Number of decimal places of output. Default is 0.

Value

A character string of time difference.

Examples

## Not run:

t0 = Sys.time()
dtime(t0)

## End(Not run)

EFA

Principal Component Analysis (PCA) and Exploratory Factor analysis (EFA).

Description

An extension of psych::principal() and psych::fa(), performing either Principal Component Analysis (PCA) or Exploratory Factor Analysis (EFA).

Three options to specify variables:

1. var + items: use the common and unique parts of variable names.
2. vars: directly define a character vector of variables.
3. varrange: use the starting and stopping positions of variables.
Usage

EFA(
  data,
  var,
  items,
  vars = NULL,
  varrange = NULL,
  rev = NULL,
  method = c("pca", "pa", "ml", "minres", "uls", "ols", "wls", "gls", "alpha"),
  rotation = c("none", "varimax", "oblimin", "promax", "quartimax", "equamax"),
  nfactors = c("eigen", "parallel", ":0")
)

PCA(..., method = "pca")

Arguments

data            Data frame.
var             [Option 1] The common part across the variables. e.g., "RSES"
items           [Option 1] The unique part across the variables. e.g., 1:10
vars            [Option 2] A character vector specifying the variables. e.g., c("X1", "X2", "X3", "X4", "X5")
varrange        [Option 3] A character string specifying the positions ("starting:stopping") of variables. e.g., "A1:E5"
rev             [Optional] Variables that need to be reversed. It can be (1) a character vector specifying the reverse-scoring variables (recommended), or (2) a numeric vector specifying the item number of reverse-scoring variables (not recommended).
method          Extraction method.
  • "pca" - Principal Component Analysis (default)
  • "pa" - Principal Axis Factor Analysis
  • "ml" - Maximum Likelihood Factor Analysis
  • "minres" - Minimum Residual Factor Analysis
  • "uls" - Unweighted Least Squares Factor Analysis
  • "ols" - Ordinary Least Squares Factor Analysis
  • "wls" - Weighted Least Squares Factor Analysis
  • "gls" - Generalized Least Squares Factor Analysis
  • "alpha" - Alpha Factor Analysis (Kaiser & Coffey, 1965)
rotation Rotation method.
  • "none" - None (not suggested)
  • "varimax" - Varimax (default)
  • "oblimin" - Direct Oblimin
  • "promax" - Promax
  • "quartimax" - Quartimax
  • "equamax" - Equamax

nfactors How to determine the number of factors/components?
  • "eigen" - based on eigenvalue (> minimum eigenvalue) (default)
  • "parallel" - based on parallel analysis
  • (any number >= 1) - user-defined fixed number

sort.loadings Sort factor/component loadings by size? Default is TRUE.
hide.loadings A number (0~1) for hiding absolute factor/component loadings below this value. Default is 0 (does not hide any loading).
plot.scree Display the scree plot? Default is TRUE.
kaiser Do the Kaiser normalization (as in SPSS)? Default is TRUE.
max.iter Maximum number of iterations for convergence. Default is 25 (the same as in SPSS).
min.eigen Minimum eigenvalue (used if nfactors="eigen"). Default is 1.
digits, nsmall Number of decimal places of output. Default is 3.
file File name of MS Word (.doc).
... Arguments passed from PCA() to EFA().

Value
A list of results:
  result The R object returned from psych::principal() or psych::fa()
  result.kaiser The R object returned from psych::kaiser() (if any)
  extraction.method Extraction method
  rotation.method Rotation method
  eigenvalues A data.frame of eigenvalues and sum of squared (SS) loadings
  loadings A data.frame of factor/component loadings and communalities
  scree.plot A ggplot2 object of the scree plot

Functions
  • EFA: Exploratory Factor Analysis
  • PCA: Principal Component Analysis - a wrapper of EFA(., method="pca")

Note
Results based on the varimax rotation method are identical to SPSS. The other rotation methods may produce results slightly different from SPSS.
EMMEANS

Simple-effect analysis and post-hoc multiple comparison.

Description

Perform (1) simple-effect (and simple-simple-effect) analyses, including both simple main effects and simple interaction effects, and (2) post-hoc multiple comparisons (e.g., pairwise, sequential, polynomial), with p values adjusted for factors with >= 3 levels.

This function is based on and extends (1) emmeans::joint_tests(), (2) emmeans::emmeans(), and (3) emmeans::contrast(). You only need to specify the model object, to-be-tested effect(s), and moderator(s). Almost all results you need will be displayed together, including effect sizes (partial \(\eta^2\) and Cohen’s d) and their confidence intervals (CIs). 90% CIs for partial \(\eta^2\) and 95% CIs for Cohen’s d are reported.

By default, the root mean square error (RMSE) is used to compute the pooled SD for Cohen’s d. Specifically, it uses:

1. the square root of mean square error (MSE) for between-subjects designs;
2. the square root of mean variance of all paired differences of the residuals of repeated measures for within-subjects and mixed designs.

In both situations, it extracts the lm object from the returned value of MANOVA(). Then, it mainly uses sigma() and residuals(), respectively, to do these estimates. For source code, please see the file bruceR_stats_03_manova.R on the GitHub Repository.
Disclaimer: There is substantial disagreement on the appropriate pooled $SD$ to use in computing the effect size. For alternative methods, see `emmeans::eff_size()` and `effectsize::t_to_d()`. Users should not take the default output as the only right results and are completely responsible for specifying `sd.pooled`.

Usage

```r
EMMEANS(
  model,
  effect = NULL,
  by = NULL,
  contrast = "pairwise",
  reverse = TRUE,
  p.adjust = "bonferroni",
  sd.pooled = NULL,
  model.type = "multivariate",
  digits = 3,
  nsmall = digits
)
```

Arguments

- **model**: The model object returned by `MANOVA`.
- **effect**: Effect(s) you want to test. If set to a character string (e.g., "A"), it reports the results of omnibus test or simple main effect. If set to a character vector (e.g., `c("A","B")`), it also reports the results of simple interaction effect.
- **by**: Moderator variable(s). Default is `NULL`.
- **contrast**: Contrast method for multiple comparisons. Default is "pairwise". Alternatives can be "pairwise" ("revpairwise"), "seq" ("consec"), "poly", "eff". For details, see `?emmeans::contrast-methods`.
- **reverse**: The order of levels to be contrasted. Default is `TRUE` (higher level vs. lower level).
- **p.adjust**: Adjustment method of $p$ values for multiple comparisons. Default is "bonferroni". For polynomial contrasts, default is "none". Alternatives can be "none", "fdr", "hochberg", "hommel", "holm", "tukey", "mvt", "dunnetttx", "sidak", "scheffe", "bonferroni". For details, see `stats::p.adjust()` and `emmeans::summary()`.
- **sd.pooled**: By default, it uses $\sqrt{\text{MSE}}$ (root mean square error, RMSE) as the pooled $SD$ to compute Cohen’s $d$. Users may specify this argument as the $SD$ of a reference group, or use `effectsize::sd_pooled()` to obtain a pooled $SD$. For an issue about the computation method of Cohen’s $d$, see Disclaimer above.
- **model.type**: "multivariate" returns the results of pairwise comparisons identical to SPSS, which uses the `lm` (rather than `aov`) object of the model for `emmeans::joint_tests()` and `emmeans::emmeans()`. "univariate" requires also specifying `aov.include=TRUE` in `MANOVA` (not recommended by the afex package; for details, see `afex::aov_ez()`).
- **digits, nsmall**: Number of decimal places of output. Default is 3.
Value

The same model object as returned by `MANOVA` (for recursive use), along with a list of tables: `sim` (simple effects), `emm` (estimated marginal means), `con` (contrasts).

Each `EMMEANS(...)` appends one list to the returned object.

Interaction Plot

You can save the returned object and use the `emmeans::emmip()` function to create an interaction plot (based on the fitted model and a formula specification). For usage, please see the help page of `emmeans::emmip()`. It returns an object of class `ggplot`, which can be easily modified and saved using `ggplot2` syntax.

Statistical Details

Some may confuse the statistical terms "simple effects", "post-hoc tests", and "multiple comparisons". Such a confusion is not uncommon. Here I explain what these terms actually refer to.

1. **Simple Effect** When we speak of "simple effect", we are referring to ...
   
   - simple main effect
   - simple interaction effect (only for designs with 3 or more factors)
   - simple simple effect (only for designs with 3 or more factors)

   When the interaction effect in ANOVA is significant, we should then perform a "simple-effect analysis". In regression, we call this "simple-slope analysis". They are identical in statistical principles.

   In a two-factors design, we only test "**simple main effect**". That is, at different levels of a factor "B", the main effects of "A" would be different. However, in a three-factors (or more) design, we may also test "**simple interaction effect**" and "**simple simple effect**". That is, at different combinations of levels of factors "B" and "C", the main effects of "A" would be different.

   To note, simple effects *per se* never require *p*-value adjustment, because what we test in simple-effect analyses are still "omnibus F-tests".

2. **Post-Hoc Test** The term "post-hoc" means that the tests are performed after ANOVA. Given this, some may (wrongly) regard simple-effect analyses also as a kind of post-hoc tests. However, these two terms should be distinguished. In many situations, "post-hoc tests" only refer to "**post-hoc comparisons**" using t-tests and some *p*-value adjustment techniques. We need post-hoc comparisons *only when there are factors with 3 or more levels*.

   Post-hoc tests are totally *independent of* whether there is a significant interaction effect. It *only deals with factors with multiple levels*. In most cases, we use pairwise comparisons to do post-hoc tests. See the next part for details.

3. **Multiple Comparison** As mentioned above, multiple comparisons are indeed post-hoc tests but have no relationship with simple-effect analyses. Post-hoc multiple comparisons are *independent of* interaction effects and simple effects. Furthermore, if a simple main effect contains 3 or more levels, we also need to do multiple comparisons *within* the simple-effect analysis.

   In this situation, we also need *p*-value adjustment with methods such as Bonferroni, Tukey’s HSD (honest significant difference), FDR (false discovery rate), and so forth.

   Options for multiple comparison:
EMMEANS

- "pairwise" - Pairwise comparisons (default is "higher level - lower level")
- "seq" or "consec" - Consecutive (sequential) comparisons
- "poly" - Polynomial contrasts (linear, quadratic, cubic, quartic, ...)
- "eff" - Effect contrasts (vs. the grand mean)

See Also
TTEST, MANOVA, bruceR-demodata

Examples

#### Between-Subjects Design ####

```r
between.1
MANOVA(between.1, dv="SCORE", between="A") %>%
  EMMEANS("A")

MANOVA(between.1, dv="SCORE", between="A") %>%
  EMMEANS("A", p.adjust="tukey")

MANOVA(between.1, dv="SCORE", between="A") %>%
  EMMEANS("A", contrast="seq")

MANOVA(between.1, dv="SCORE", between="A") %>%
  EMMEANS("A", contrast="poly")
```

```r
between.2
MANOVA(between.2, dv="SCORE", between=c("A", "B")) %>%
  EMMEANS("A", by="B") %>%
  EMMEANS("B", by="A")

## How to create an interaction plot using `emmeans::emmip()`?
## See help page for its usage: ?emmeans::emmip()

m = MANOVA(between.2, dv="SCORE", between=c("A", "B"))
emmip(m, ~ A | B, CIs=TRUE)
emmip(m, ~ B | A, CIs=TRUE)
emmip(m, B ~ A, CIs=TRUE)
emmip(m, A ~ B, CIs=TRUE)
```

#### Within-Subjects Design ####

```r
within.1
MANOVA(within.1, dv="A1:A4", dvs.pattern="A(.)",
  within="A") %>%
  EMMEANS("A")
```

within.2
EMMEANS

MANOVA(within.2, dvs="A1B1:A2B3", dvs.pattern="A(.)B(.)",
   within=c("A", "B")) %>%
   EMMEANS("A", by="B") %>%
   EMMEANS("B", by="A") # singular error matrix
# :::::::::::::::::::::::::::::::::::::::::::
# This would produce a WARNING because of
# the linear dependence of A2B2 and A2B3.
# See: Corr(within.2[c("A2B2", "A2B3")])

within.3
   within=c("A", "B", "C")) %>%
   EMMEANS("A", by="B") %>%
   EMMEANS(c("A", "B"), by="C") %>%
   EMMEANS("A", by=c("B", "C"))
## Just to name a few...
## You may test other combinations...

#### Mixed Design ####

mixed.2_1b1w
MANOVA(mixed.2_1b1w, dvs="B1:B3", dvs.pattern="B(.)",
   between="A", within="B", sph.correction="GG") %>%
   EMMEANS("A", by="B") %>%
   EMMEANS("B", by="A")

mixed.3_1b2w
MANOVA(mixed.3_1b2w, dvs="B1C1:B2C2", dvs.pattern="B(.)C(.)",
   between="A", within=c("B", "C")) %>%
   EMMEANS("A", by="B") %>%
   EMMEANS(c("A", "B"), by="C") %>%
   EMMEANS("A", by=c("B", "C"))
## Just to name a few...
## You may test other combinations...

mixed.3_2b1w
MANOVA(mixed.3_2b1w, dvs="B1:B2", dvs.pattern="B(.)",
   between=c("A", "C"), within="B") %>%
   EMMEANS("A", by="B") %>%
   EMMEANS("A", by="C") %>%
   EMMEANS(c("A", "B"), by="C") %>%
   EMMEANS("B", by=c("A", "C"))
## Just to name a few...
## You may test other combinations...

#### Other Examples ####

air = airquality
air$Day.1or2 = ifelse(air$Day %% 2 == 1, 1, 2) %>%
   factor(levels=1:2, labels=c("odd", "even"))
MANOVA(air, dv="Temp", between=c("Month", "Day.1or2"),
covariate=c("Solar.R", "Wind");
EMMEANS("Month", contrast="seq");
EMMEANS("Month", by="Day.1or2", contrast="poly")

---

export

Export data to a file (TXT, CSV, Excel, SPSS, Stata, ...) or clipboard.

Description

Export data to a file, with format automatically judged from file extension. This function is inspired by rio::export() and has several modifications. Its purpose is to avoid using lots of write_xxx() functions in your code and to provide one tidy function for data export.

It supports many file formats and uses corresponding R functions:

- Plain text (.txt, .csv, .csv2, .tsv, .psv), using data.table::fwrite(); if the encoding argument is specified, using utils::write.table() instead
- Excel (.xls, .xlsx), using openxlsx::write.xlsx()
- SPSS (.sav), using haven::write_sav()
- Stata (.dta), using haven::write_dta()
- R objects (.rda, .rdata, .Rdata), using base::save()
- R serialized objects (.rds), using base::saveRDS()
- Clipboard (on Windows and Mac OS), using clipr::write_clip()
- Other formats, using rio::export()

Usage

export(
  x,
  file,
  sheet = NULL,
  encoding = NULL,
  header = "auto",
  overwrite = TRUE
)

Arguments

x Any R object, usually a data frame (data.frame, data.table, tbl_df). Multiple R objects should be included in a named list (see examples).
If you want to save R objects other than a data frame (e.g., model results), you’d better specify file with extensions .rda, .rdata, or .Rdata.

covariate=c("Solar.R", "Wind")
EMMEANS("Month", contrast="seq")
EMMEANS("Month", by="Day.1or2", contrast="poly")
export

sheet [Only for Excel] Excel sheet name(s). Default is Sheet1, Sheet2, ... You may specify multiple sheet names in a character vector `c()` with the same length as `x` (see examples).

encoding File encoding. Default is NULL. Alternatives can be "UTF-8", "GBK", "CP936", etc. If you find messy code for Chinese text in the exported data (often in CSV when opened with Excel), it is usually effective to set encoding="GBK" or encoding="CP936".

header Does the first row contain column names (TRUE or FALSE)? Default is "auto".

overwrite Overwrite the existing file (if any)? Default is TRUE.

Value
No return value.

See Also
import, print_table

Examples

## Not run:

```r
export(airquality) # paste to clipboard
export(airquality, file="mydata.csv")
export(airquality, file="mydata.sav")

export(list(airquality, npk), file="mydata.xlsx") # Sheet1, Sheet2
export(list(air=airquality, npk=npk), file="mydata.xlsx") # a named list
export(list(air=airquality, npk), sheet=c("air", "npk"), file="mydata.xlsx")

export(list(a=1, b=npk, c="character"), file="abc.Rdata") # .rda, .rdata
d = import("abc.Rdata") # load only the first object and rename it to 'd'
load("abc.Rdata") # load all objects with original names to environment

export(lm(yield ~ N*P*K, data=npk), file="lm_npk.Rdata")
model = import("lm_npk.Rdata")
load("lm_npk.Rdata") # because x is unnamed, the object has a name "List1"

export(list(m1=lm(yield ~ N*P*K, data=npk)), file="lm_npk.Rdata")
model = import("lm_npk.Rdata")
load("lm_npk.Rdata") # because x is named, the object has a name "m1"

## End(Not run)
```
formatF

Format numeric values.

Description
Format numeric values.

Usage
formatF(x, digits = 3, nsmall = digits)

Arguments
x A number or numeric vector.
digits, nsmall Number of decimal places of output. Default is 3.

Value
Formatted character string.

See Also
format, formatN

Examples
formatF(pi, 20)

formatN
Format "1234" to "1,234".

Description
Format "1234" to "1,234".

Usage
formatN(x, mark = ",")

Arguments
x A number or numeric vector.
mark Usually ",".
Value

Formatted character string.

See Also

format, formatF

Examples

formula_expand(1234)

formula_expand(formula = y ~ a*b*c)

formula_expand("y ~ a*b*c")
**formula_paste**  
*Paste a formula into a string.*

**Description**
Paste a formula into a string.

**Usage**
```
formula_paste(formula)
```

**Arguments**
- **formula**: R formula.

**Value**
A character string indicating the formula.

**Examples**
```
formula_paste(y ~ x)
formula_paste(y ~ x + (1 | g))
```

---

**Freq**  
*Frequency statistics.*

**Description**
Frequency statistics.

**Usage**
```
Freq(x, varname, labels, sort = "", digits = 1, nsmall = digits, file = NULL)
```

**Arguments**
- **x**: A vector of values (or a data frame).
- **varname**: [Optional] Variable name, if x is a data frame.
- **labels**: [Optional] A vector re-defining the labels of values.
- **sort**: "" (default, sorted by the order of variable values/labels), "-" (decreasing by N), or "+" (increasing by N).
- **digits, nsmall**: Number of decimal places of output. Default is 1.
- **file**: File name of MS Word (.doc).
Value

A data frame of frequency statistics.

Examples

data = psych::bfi

## Input `data$variable`
Freq(data$education)
Freq(data$gender, labels=c("Male", "Female"))
Freq(data$age)

## Input one data frame and one variable name
Freq(data, "education")
Freq(data, "gender", labels=c("Male", "Female"))
Freq(data, "age")

GLM_summary

Tidy report of GLM (lm and glm models).

Description

NOTE: model_summary is preferred.

Usage

GLM_summary(
  model,  # A model fitted with lm or glm function.
  robust = FALSE,  # [Only for lm and glm] FALSE (default), TRUE (then the default is "HC1"), "HC0", "HC1", "HC2", "HC3", "HC4", "HC4m", or "HC5". It will add a table with heteroskedasticity-robust standard errors (aka. Huber-White standard errors). For details, see ?sandwich::vcovHC and ?jtools::summ.lm.
  cluster = NULL,  # [Only for lm and glm] Cluster-robust standard errors are computed if cluster is set to the name of the input data’s cluster variable or is a vector of clusters.
  digits = 3,  # [Only for lm and glm] Cluster-robust standard errors are computed if cluster is set to the name of the input data’s cluster variable or is a vector of clusters.
  nsmall = digits,
  ...
)

Arguments

model  # A model fitted with lm or glm function.
robust  # [Only for lm and glm] FALSE (default), TRUE (then the default is "HC1"), "HC0", "HC1", "HC2", "HC3", "HC4", "HC4m", or "HC5". It will add a table with heteroskedasticity-robust standard errors (aka. Huber-White standard errors). For details, see ?sandwich::vcovHC and ?jtools::summ.lm.
cluster  # [Only for lm and glm] Cluster-robust standard errors are computed if cluster is set to the name of the input data’s cluster variable or is a vector of clusters.
grand_mean_center

digits, nsmall Number of decimal places of output. Default is 3.
... Other arguments. You may re-define formula, data, or family.

Value

No return value.

See Also

print_table (print simple table)
model_summary (highly suggested)
HLM_summary
regress

Examples

## Example 1: OLS regression
lm = lm(Temp ~ Month + Day + Wind + Solar.R, data=airquality)
GLM_summary(lm)
GLM_summary(lm, robust="HC1")
# Stata's default is "HC1"
# R package <sandwich>'s default is "HC3"

## Example 2: Logistic regression
glm = glm(case ~ age + parity + education + spontaneous + induced,
data=infert, family=binomial)
GLM_summary(glm)
GLM_summary(glm, robust="HC1", cluster="stratum")

---

grand_mean_center \hspace{1cm} Grand-mean centering.

Description

Compute grand-mean centered variables. Usually used for GLM interaction-term predictors and HLM level-2 predictors.

Usage

grand_mean_center(data, vars = names(data), std = FALSE, add.suffix = "")

Arguments

data \hspace{1cm} Data object.
vars \hspace{1cm} Variable(s) to be centered.
std \hspace{1cm} Standardized or not. Default is FALSE.
add.suffix \hspace{1cm} The suffix of the centered variable(s). Default is ".". You may set it to ",_c", ",_center", etc.
Value

A new data object containing the centered variable(s).

See Also

group_mean_center

Examples

d = data.table(a=1:5, b=6:10)
d.c = grand_mean_center(d, "a")
d.c

d.c = grand_mean_center(d, c("a", "b"), add.suffix="_center")
d.c

granger_causality

Granger causality test (multivariate).

Description

Granger test of predictive causality (between multivariate time series) based on vector autoregression (VAR) model. Its output resembles the output of the vargranger command in Stata (but here using an $F$ test).

Usage

granger_causality(
  varmodel,
  var.y = NULL,
  var.x = NULL,
  test = c("F", "Chisq"),
  file = NULL,
  check.dropped = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>varmodel</td>
<td>VAR model fitted using the <code>vars::VAR()</code> function.</td>
</tr>
<tr>
<td>var.y, var.x</td>
<td>[optional] Default is NULL (all variables). If specified, then perform tests for specific variables. Values can be a single variable (e.g., &quot;X&quot;), a vector of variables (e.g., c(&quot;X1&quot;, &quot;X2&quot;)), or a string containing regular expression (e.g., &quot;X1</td>
</tr>
<tr>
<td>test</td>
<td>$F$ test and/or Wald $\chi^2$ test. Default is both: c(&quot;F&quot;, &quot;Chisq&quot;).</td>
</tr>
<tr>
<td>file</td>
<td>File name of MS Word (.doc).</td>
</tr>
<tr>
<td>check.dropped</td>
<td>Check dropped variables. Default is FALSE.</td>
</tr>
</tbody>
</table>
Granger causality test (based on VAR model) examines whether the lagged values of a predictor (or predictors) help to predict an outcome when controlling for the lagged values of the outcome itself. Granger causality does not necessarily constitute a true causal effect.

Value
A data frame of results.

See Also
ccf_plot, granger_test

Examples
```r
## Not run:
# R package "vars" should be installed
library(vars)
data(Canada)
VARselect(Canada)
vm = VAR(Canada, p=3)
model_summary(vm)
granger_causality(vm)
## End(Not run)
```

---

**granger_test**  
*Granger causality test (bivariate).*

**Description**
Granger test of predictive causality (between two time series) using the `lmtest::grangertest()` function.

**Usage**
granger_test(formula, data, lags = 1:5, test.reverse = TRUE, file = NULL)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>Model formula like <code>y ~ x</code>.</td>
</tr>
<tr>
<td>data</td>
<td>Data frame.</td>
</tr>
<tr>
<td>lags</td>
<td>Time lags. Default is <code>1:5</code>.</td>
</tr>
<tr>
<td>test.reverse</td>
<td>Whether to test reverse causality. Default is <code>TRUE</code>.</td>
</tr>
<tr>
<td>file</td>
<td>File name of MS Word (.doc).</td>
</tr>
</tbody>
</table>
Details

Granger causality test examines whether the lagged values of a predictor have an incremental role in predicting (i.e., help to predict) an outcome when controlling for the lagged values of the outcome. Granger causality does not necessarily constitute a true causal effect.

Value

A data frame of results.

See Also

ccf_plot, granger_causality

Examples

granger_test(chicken ~ egg, data=lmtest::ChickEgg)
granger_test(chicken ~ egg, data=lmtest::ChickEgg, lags=1:10, file="Granger.doc")
unlink("Granger.doc")  # delete file for code check

---

group_mean_center  Group-mean centering.

Description

Compute group-mean centered variables. Usually used for HLM level-1 predictors.

Usage

```r
group_mean_center(
  data, 
  vars = setdiff(names(data), by), 
  by, 
  std = FALSE, 
  add.suffix = "", 
  add.group.mean = "_mean"
)
```

Arguments

data  Data object.
vars  Variable(s) to be centered.
by  Grouping variable.
std  Standardized or not. Default is FALSE.
add.suffix  The suffix of the centered variable(s). Default is ".". You may set it to ".c", ".center", etc.
add.group.mean  The suffix of the variable name(s) of group means. Default is ".mean" (see Examples).
HLM_ICC_rWG

Value

A new data object containing the centered variable(s).

See Also

group_mean_center

Examples

d = data.table(x=1:9, g=rep(1:3, each=3))
d.c = group_mean_center(d, "x", by="g")
d.c

d.c = group_mean_center(d, "x", by="g", add.suffix="_c")
d.c

HLM_ICC_rWG  Tidy report of HLM indices: ICC(1), ICC(2), and rWG/rWG(J).

Description

Compute ICC(1) (non-independence of data), ICC(2) (reliability of group means), and rWG/rWG(J) (within-group agreement for single-item/multi-item measures) in multilevel analysis (HLM).

Usage

HLM_ICC_rWG(
  data,
  group,
  icc.var,
  rwg.vars = icc.var,
  rwg.levels = 0,
  digits = 3,
  nsmall = digits
)

Arguments

data  Data frame.
group  Grouping variable.
icc.var  Key variable for analysis (usually the dependent variable).
rgw.vars  Default is icc.var. It can be:
  • A single variable (single-item measure), then computing rWG.
  • Multiple variables (multi-item measure), then computing rWG(J), where J = the number of items.
**rwg.levels**

As rWG/rWG(J) compares the actual group variance to the expected random variance (i.e., the variance of uniform distribution, $\sigma_E U^2$), it is required to specify which type of uniform distribution is.

- For **continuous** uniform distribution, $\sigma_E U^2 = (\text{max} - \text{min})^2 / 12$. Then `rwg.levels` is not useful and will be set to 0 (the default).
- For **discrete** uniform distribution, $\sigma_E U^2 = (A^2 - 1) / 12$, where $A$ is the number of response options (levels). Then `rwg.levels` should be provided (= $A$ in the above formula). For example, if the measure is a 5-point Likert scale, you should set `rwg.levels=5`.

**digits, nsmall**

Number of decimal places of output. Default is 3.

**Details**

**ICC(1) (intra-class correlation, or non-independence of data)**

$\text{ICC}(1) = \frac{\text{var}.u0}{(\text{var}.u0 + \text{var}.e)} = \frac{\sigma^2_{u0}}{\left(\sigma^2_{u0} + \sigma^2_e\right)}$

ICC(1) is the ICC we often compute and report in multilevel analysis (usually in the Null Model, where only the random intercept of group is included). It can be interpreted as either "the proportion of variance explained by groups" (i.e., heterogeneity between groups) or "the expectation of correlation coefficient between any two observations within any group" (i.e., homogeneity within groups).

**ICC(2) (reliability of group means)**

$\text{ICC}(2) = \frac{\text{mean} (\text{var}.u0 / (\text{var}.u0 + \text{var}.e / n_k)) = \Sigma [\sigma^2_{u0} / (\sigma^2_{u0} + \sigma^2_e / n_k)] / K}$

ICC(2) is a measure of "the representativeness of group-level aggregated means for within-group individual values" or "the degree to which an individual score can be considered a reliable assessment of a group-level construct".

**rWG/rWG(J) (within-group agreement for single-item/multi-item measures)**

$rWG = 1 - \frac{\sigma^2_e}{\sigma^2_E U}$

$rWG(J) = 1 - \left(\frac{\sigma^2_{M,J}}{\sigma^2_E U}\right) / [J * (1 - \frac{\sigma^2_{M,J}}{\sigma^2_E U}) + \frac{\sigma^2_{M,J}}{\sigma^2_E U}]$

$rWG/rWG(J)$ is a measure of within-group agreement or consensus. Each group has an `rWG/rWG(J)`.

* **Note for the above formulas**
  - $\sigma^2_{u0}$: between-group variance (i.e., tau00)
  - $\sigma^2_e$: within-group variance (i.e., residual variance)
  - $n_k$: group size of the k-th group
  - $K$: number of groups
  - $\sigma^2_{M,J}$: actual group variance of the k-th group
  - $\sigma^2_{M,J}$: mean value of actual group variance of the k-th group across all J items
  - $\sigma^2_E U$: expected random variance (i.e., the variance of uniform distribution)
  - $J$: number of items

**Value**

Invisibly return a list of results.

**References**


See Also

R package "multilevel"

Examples

```r
data = lme4::sleepstudy  # continuous variable
HLM_ICC_rWG(data, group="Subject", icc.var="Reaction")

data = lmerTest::carrots  # 7-point scale
HLM_ICC_rWG(data, group="Consumer", icc.var="Preference", rwg.vars="Preference", rwg.levels=7)
HLM_ICC_rWG(data, group="Consumer", icc.var="Preference", rwg.vars=c("Sweetness", "Bitter", "Crisp"), rwg.levels=7)
```

---

**HLM_summary**  
*Tidy report of HLM (lmer and glmer models).*

**Description**

NOTE: *model_summary* is preferred.

**Usage**

`HLM_summary(model = NULL, test.rand = FALSE, digits = 3, nsmall = digits, ...)`

**Arguments**

- `model`: A model fitted with lmer or glmer function using the lmerTest package.
- `test.rand`: [Only for lmer and glmer] TRUE or FALSE (default). Test random effects (i.e., variance components) by using the likelihood-ratio test (LRT), which is asymptotically chi-square distributed. For large datasets, it is much time-consuming.
- `digits, nsmall`: Number of decimal places of output. Default is 3.
- `...`: Other arguments. You may re-define formula, data, or family.

**Value**

No return value.
References


See Also

`print_table` (print simple table)

`model_summary` (highly suggested)

`GLM_summary`

`regress`

Examples

```r
library(lmerTest)

## Example 1: data from lme4::sleepstudy
# (1) 'Subject' is a grouping/clustering variable
# (2) 'Days' is a level-1 predictor nested within 'Subject'
# (3) No level-2 predictors
m1 = lmer(Reaction ~ (1 | Subject), data=sleepstudy)
m2 = lmer(Reaction ~ Days + (1 | Subject), data=sleepstudy)
m3 = lmer(Reaction ~ Days + (Days | Subject), data=sleepstudy)
HLM_summary(m1)
HLM_summary(m2)
HLM_summary(m3)

## Example 2: data from lmerTest::carrots
# (1) 'Consumer' is a grouping/clustering variable
# (2) 'Sweetness' is a level-1 predictor
# (3) 'Age' and 'Frequency' are level-2 predictors
hlm.1 = lmer(Preference ~ Sweetness + Age + Frequency +
             (1 | Consumer), data=carrots)
hlm.2 = lmer(Preference ~ Sweetness + Age + Frequency +
             (Sweetness | Consumer) + (1 | Product), data=carrots)
HLM_summary(hlm.1)
HLM_summary(hlm.2)
```
import

**Description**

Import data from a file, with format automatically judged from file extension. This function is inspired by `rio::import()` and has several modifications. Its purpose is to avoid using lots of `read_xxx()` functions in your code and to provide one tidy function for data import.

It supports many file formats and uses corresponding R functions:

- Plain text (.txt, .csv, .csv2, .tsv, .psv), using `data.table::fread()`
- Excel (.xls, .xlsx), using `readxl::read_excel()`
- SPSS (.sav), using `foreign::read.spss()`; if failed, using `haven::read_sav()` instead
- Stata (.dta), using `foreign::read.dta()`; if failed, using `haven::read_dta()` instead
- R objects (.rda, .rdata, .Rdata), using `base::load()`
- R serialized objects (.rds), using `base::readRDS()`
- Clipboard (on Windows and Mac OS), using `clipr::read_clip_tbl()`
- Other formats, using `rio::import()`

**Usage**

```r
import(
  file,
  sheet = NULL,
  range = NULL,
  encoding = NULL,
  header = "auto",
  setclass = as,
  as = "data.frame"
)
```

**Arguments**

- `file` File name (with extension). If unspecified, then data will be imported from clipboard.
- `sheet` [Only for Excel] Excel sheet name (or sheet number). Default is the first sheet. Ignored if the sheet is specified via range.
- `range` [Only for Excel] Excel cell range. Default are all cells in a sheet. You may specify it as `range= "A1:E100"` or `range = "Sheet1!A1:E100"`.
- `encoding` File encoding. Default is NULL. Alternatives can be "UTF-8", "GBK", "CP936", etc.
  If you find messy code for Chinese text in the imported data, it is usually effective to set `encoding= "UTF-8"`.
- `header` Does the first row contain column names (TRUE or FALSE)? Default is "auto".
- `setclass, as` Class of the imported data. Default is "data.frame". Ignored if the data file is R object (.rds, .rda, .rdata, .Rdata).
  Alternatives can be:
  - `data.frame`: "data.frame", "df", "DF"
  - `data.table`: "data.table", "dt", "DT"
  - `tbl_df`: "tibble", "tbl_df", "tbl"
Value

A data object (default class is data.frame).

See Also

export

Examples

## Not run:

# Import data from system clipboard
data = import() # read from clipboard (on Windows and Mac OS)

# If you have an Excel file named "mydata.xlsx"
export(airquality, file="mydata.xlsx")

# Import data from a file
data = import("mydata.xlsx") # default: data.frame
data = import("mydata.xlsx", as="data.table")

## End(Not run)
### Arguments

**lavaan**  
Model object fitted by **lavaan**.

**ci**  
Method for estimating standard error (SE) and 95% confidence interval (CI). Default is "raw" (the standard approach of lavaan). Other options:

- "boot" Percentile Bootstrap
- "bc.boot" Bias-Corrected Percentile Bootstrap
- "bca.boot" Bias-Corrected and Accelerated (BCa) Percentile Bootstrap

**nsim**  
Number of simulation samples (bootstrap resampling) for estimating SE and 95% CI. In formal analyses, **nsim=1000 (or larger)** is strongly suggested.

**seed**  
Random seed for obtaining reproducible results. Default is NULL.

**digits, nsmall**  
Number of decimal places of output. Default is 3.

**print**  
Print results. Default is TRUE.

**covariance**  
Print (co)variances. Default is FALSE.

**file**  
File name of MS Word (.doc).

### Value

Invisibly return a list of results:

- **fit** Model fit indices.
- **measure** Latent variable measures.
- **regression** Regression paths.
- **covariance** Variances and/or covariances.
- **effect** Defined effect estimates.

### See Also

PROCESS, CFA

### Examples

```r
## Simple Mediation:
## Solar.R (X) => Ozone (M) => Temp (Y)

# PROCESS(airquality, y="Temp", x="Solar.R",
# meds="Ozone", ci="boot", nsim=1000, seed=1)

model = "
Ozone ~ a*Solar.R
Temp ~ c.*Solar.R + b*Ozone
Indirect := a*b
Direct := c.
Total := c. + a*b
"
lt = lavaan::sem(model=model, data=airquality)
lavaan::summary(lt, fit.measure=TRUE, ci=TRUE, nd=3)  # raw output
```
lavaan_summary(lv)
# lavaan_summary(lv, ci="boot", nsim=1000, seed=1)

## Serial Multiple Mediation:
## Solar.R (X) => Ozone (M1) => Wind (M2) => Temp (Y)

# PROCESS(airquality, y="Temp", x="Solar.R",
# meds=c("Ozone", "Wind"),
# med.type="serial", ci="boot", nsim=1000, seed=1)

model0 = 
Ozone ~ a1*Solar.R
Wind ~ a2*Solar.R + d12*Ozone
Temp ~ c.*Solar.R + b1*Ozone + b2*Wind
Indirect_All := a1*b1 + a2*b2 + a1*d12*b2
Ind_X_M1_Y := a1*b1
Ind_X_M2_Y := a2*b2
Ind_X_M1_M2_Y := a1*d12*b2
Direct := c.
Total := c. + a1*b1 + a2*b2 + a1*d12*b2
"

lv0 = lavaan::sem(model=model0, data=airquality)
lavaan::summary(lv0, fit.measure=TRUE, ci=TRUE, nd=3) # raw output
lavaan_summary(lv0)
# lavaan_summary(lv0, ci="boot", nsim=1000, seed=1)

model1 = 
Ozone ~ a1*Solar.R
Wind ~ d12*Ozone
Temp ~ c.*Solar.R + b1*Ozone + b2*Wind
Indirect_All := a1*b1 + a1*d12*b2
Ind_X_M1_Y := a1*b1
Ind_X_M1_M2_Y := a1*d12*b2
Direct := c.
Total := c. + a1*b1 + a1*d12*b2
"

lv1 = lavaan::sem(model=model1, data=airquality)
lavaan::summary(lv1, fit.measure=TRUE, ci=TRUE, nd=3) # raw output
lavaan_summary(lv1)
# lavaan_summary(lv1, ci="boot", nsim=1000, seed=1)

LOOKUP

Search, match, and look up values (like Excel’s functions INDEX + MATCH).

Description

In Excel, we can use VLOOKUP, HLOOKUP, XLOOKUP (a new function released in 2019), or the combination of INDEX and MATCH to search, match, and look up values. Here I provide a similar function.
LOOKUP

Usage

LOOKUP(
  data,
  vars,
  data.ref,
  vars.ref,
  vars.lookup,
  return = c("new.data", "new.var", "new.value")
)

Arguments

data       Main data.
vars       Character (vector), specifying the variable(s) to be searched in data.
data.ref    Reference data containing both the reference variable(s) and the lookup variable(s).
vars.ref    Character (vector), with the same length and order as vars, specifying the reference variable(s) to be matched in data.ref.
vars.lookup Character (vector), specifying the variable(s) to be looked up and returned from data.ref.
return      What to return. Default ("new.data") is to return a data frame with the lookup values added. You may also set it to "new.var" or "new.value".

Details

If multiple values were simultaneously matched, a warning message would be printed.

Value

New data object, new variable, or new value (see the argument return).

See Also

dplyr::left_join()
XLOOKUP: Excel University

Examples

ref = data.table(City=rep(c("A", "B", "C"), each=5),
                   Year=rep(2013:2017, times=3),
                   GDP=sample(1000:2000, 15),
                   PM2.5=sample(10:300, 15))

ref

data = data.table(sub=1:5,
                   city=c("A", "A", "B", "C", "C"),
data
**MANOVA**

**Multi-factor ANOVA.**

**Description**

Multi-factor ANOVA (between-subjects, within-subjects, and mixed designs), with and without covariates (ANCOVA).

This function is based on and extends `afex::aov_ez()`. You only need to specify the data, dependent variable(s), and factors (between-subjects and/or within-subjects). Almost all results you need will be displayed together, including effect sizes (partial $\eta^2$) and their confidence intervals (CIs). 90% CIs for partial $\eta^2$ (two-sided) are reported, following Steiger (2004). In addition to partial $\eta^2$, it also reports generalized $\eta^2$, following Olejnik & Algina (2003).

How to prepare your data and specify the arguments of `MANOVA`?

- **Wide-format data** (one person in one row, and repeated measures in multiple columns):
  - Between-subjects design `MANOVA(data=, dv=, between=, ...)`
  - Within-subjects design `MANOVA(data=, dvs=, dvs.pattern=, within=, ...)`
  - Mixed design `MANOVA(data=, dvs=, dvs.pattern=, between=, within=, ...)`

- **Long-format data** (one person in multiple rows, and repeated measures in one column):
  - Between-subjects design (not applicable)
  - Within-subjects design `MANOVA(data=, subID=, dv=, within=, ...)`
  - Mixed design `MANOVA(data=, subID=, dv=, between=, within=, ...)`

**Usage**

```r
MANOVA(
  data,
  subID = NULL,
  dv = NULL,
  dvs = NULL,
  dvs.pattern = NULL,
  between = NULL,
  within = NULL,
  covariate = NULL,
  ss.type = "III",
  sph.correction = "none",
  aov.include = FALSE,
  digits = 3,
  nsmall = digits,
  file = NULL
)
```

```r
LOOKUP(data, c("city", "year"), ref, c("City", "Year"), "GDP")
LOOKUP(data, c("city", "year"), ref, c("City", "Year"), c("GDP", "PM2.5"))
```
**Arguments**

- **data**  
  Data frame. Both **wide-format** and **long-format** are supported.

- **subID**  
  Subject ID (the column name). Only necessary for **long-format** data.

- **dv**  
  Dependent variable.
  - For **wide-format** data, dv only can be used for between-subjects designs. For within-subjects and mixed designs, please use dvs and dvs.pattern.
  - For **long-format** data, dv is the outcome variable.

- **dvs**  
  Repeated measures. Only for **wide-format** data (within-subjects or mixed designs).
  Two ways to specify this argument:
  - Use `:` to specify the range of variables: e.g., "A1B1:A2B3" (similar to the SPSS syntax "TO" and the order of variables matters)
  - Use a character vector to specify variable names: e.g., c("Cond1","Cond2","Cond3")

- **dvs.pattern**  
  If you use dvs, you should also specify the pattern of variable names using *regular expression*.
  Examples:
  - "Cond(.)" extracts levels from "Cond1","Cond2","Cond3",... You may rename the factor using the within argument (e.g., within="Condition")
  - "X(.+)Y(.+)" extracts levels from "X01Y01","X02Y02","XaaYbc",...
  - "X(.+)Y(.+002)" extracts levels from "X1Y1","XaYb","XaY002",...

Tips on regular expression:
  - "(.)" extracts any single character (number, letter, and other symbols)
  - "(.+)" extracts >= 1 character(s)
  - "(.*+)+" extracts >= 0 character(s)
  - "([0-9]+)" extracts any single number
  - "([a-z]+)" extracts any single letter
  - More information: Link 1 (in English) and Link 2 (in Chinese)

- **between**  
  Between-subjects factor(s). Multiple variables should be included in a character vector c().

- **within**  
  Within-subjects factor(s). Multiple variables should be included in a character vector c().

- **covariate**  
  Covariates. Multiple variables should be included in a character vector c().

- **ss.type**  
  Type of sums of squares (SS) for ANOVA. Default is "III". Possible values are "II", "III", 2, or 3.

- **sph.correction**  
  [Only for repeated measures with >= 3 levels]
  Sphericity correction method for adjusting the degrees of freedom (df) when the sphericity assumption is violated. Default is "none". If Mauchly's test of sphericity is significant, you may set it to "GG" (Greenhouse-Geisser) or "HF" (Huynh-Feldt).

- **aov.include**  
  Include the aov object in the returned object? Default is FALSE, as suggested by `afex::aov_ez()` (please see the include_aov argument in this help page, which provides a detailed explanation). If TRUE, you should also specify model.type="univariate" in EMMEANS.
digits, nsmall  Number of decimal places of output. Default is 3.
file  File name of MS Word (.doc).

Details
If observations are not uniquely identified in user-defined long-format data, the function takes averages across those multiple observations for each case. In technical details, it specifies fun_aggregate=mean in \texttt{afex::aov_ez()} and values_fn=mean in \texttt{tidyr::pivot_wider()}.

Value
A result object (list) returned by \texttt{afex::aov_ez()}, along with several other elements: between, within, data.wide, data.long.

Interaction Plot
You can save the returned object and use the \texttt{emmeans::emmip()} function to create an interaction plot (based on the fitted model and a formula specification). For usage, please see the help page of \texttt{emmeans::emmip()}. It returns an object of class \texttt{ggplot}, which can be easily modified and saved using \texttt{ggplot2} syntax.

References

See Also
TTEST, EMMEANS, bruceR-demodata

Examples

#### Between-Subjects Design ####

```r
between.1
MANOVA(between.1, dv="SCORE", between="A")
```
```r
between.2
MANOVA(between.2, dv="SCORE", between=c("A", "B"))
```
```r
between.3
MANOVA(between.3, dv="SCORE", between=c("A", "B", "C"))
```

```r
## How to create an interaction plot using 'emmeans::emmip()'?
## See help page for its usage: ?emmeans::emmip()

m = MANOVA(between.2, dv="SCORE", between=c("A", "B"))
emmp(m, ~ A | B, CIs=TRUE)
emmp(m, ~ B | A, CIs=TRUE)
emmp(m, B ~ A, CIs=TRUE)
```
```r
emmin(m, A - B, CI=TRUE)

#### Within-Subjects Design ####

within.1
MANOVA(within.1, dvs="A1:A4", dvs.pattern="A(.)", within="A")
## the same:
MANOVA(within.1, dvs=c("A1", "A2", "A3", "A4"), dvs.pattern="A(.)", within="MyFactor")  # renamed the within-subjects factor

within.2
MANOVA(within.2, dvs="A1B1:A2B3", dvs.pattern="A(.)B(.)", within=c("A", "B"))

within.3

#### Mixed Design ####

mixed.2_1b1w
MANOVA(mixed.2_1b1w, dvs="B1:B3", dvs.pattern="B(.)", between="A", within="B")
MANOVA(mixed.2_1b1w, dvs="B1:B3", dvs.pattern="B(.)", between="A", within="B", sph.correction="GG")

mixed.3_1b2w
MANOVA(mixed.3_1b2w, dvs="B1C1:B2C2", dvs.pattern="B(.)C(.)", between="A", within=c("B", "C"))

mixed.3_2b1w
MANOVA(mixed.3_2b1w, dvs="B1:B2", dvs.pattern="B(.)", between=c("A", "C"), within="B")

#### Other Examples ####

data.new = mixed.3_1b2w
names(data.new) = c("Group", "Cond_01", "Cond_02", "Cond_03", "Cond_04")
MANOVA(data.new,
  dvs="Cond_01:Cond_04",
  dvs.pattern="Cond_(..)",
  between="Group",
  within="Condition")  # rename the factor

# ?afex::obk.long
MANOVA(afex::obk.long,
  subID="id",
  dv="value",
  between=c("treatment", "gender"),
```
within=c("phase", "hour"),
cov="age",
sph.correction="GG")

med_summary

Tidy report of mediation analysis.

Description

Tidy report of mediation analysis, which is performed using the mediation package.

Usage

med_summary(model, digits = 3, nsmall = digits, file = NULL)

Arguments

model Mediation model built using mediation::mediate().
digits, nsmall Number of decimal places of output. Default is 3.
file File name of MS Word (.doc).

Value

Invisibly return a data frame containing the results.

See Also

PROCESS

Examples

## Not run:

library(mediation)
# ?mediation::mediate

## Example 1: OLS Regression
## Bias-corrected and accelerated (BCa) bootstrap confidence intervals

## Hypothesis: Solar radiation -> Ozone -> Daily temperature
lm.m = lm(Ozone ~ Solar.R + Month + Wind, data=airquality)
lm.y = lm(Temp ~ Ozone + Solar.R + Month + Wind, data=airquality)
set.seed(123) # set a random seed for reproduction
med = mediate(lm.m, lm.y,
    treat="Solar.R", mediator="Ozone",
    sims=1000, boot=TRUE, boot.ci.type="bca")
med_summary(med)
## Example 2: Multilevel Linear Model (Linear Mixed Model)
## (models must be fit using "lme4::lmer" rather than "lmerTest::lmer")
## Monte Carlo simulation (quasi-Bayesian approximation)
## (bootstrap method is not applicable to "lmer" models)

## Hypothesis: Crips -> Sweetness -> Preference (for carrots)
```r
data = lmerTest::carrots  # long-format data
data = na.omit(data)  # omit missing values
lmm.m = lme4::lmer(Sweetness ~ Crisp + Gender + Age + (1 | Consumer), data=data)
lmm.y = lme4::lmer(Preference ~ Sweetness + Crisp + Gender + Age + (1 | Consumer), data=data)
set.seed(123)  # set a random seed for reproduction
med.lmm = mediate(lmm.m, lmm.y,
                  treat="Crisp", mediator="Sweetness",
                  sims=1000)
med_summary(med.lmm)
```

## End(Not run)

---

**model_summary**

---

### Description

Tidy report of regression models (most model types are supported). This function uses:

- `texreg::screenreg()`
- `texreg::htmlreg()`
- `MuMIn::std.coef()`
- `MuMIn::r.squaredGLMM()`
- `performance::r2_mcfadden()`
- `performance::r2_nagelkerke()`

### Usage

```r
model_summary(
  model.list,
  std = FALSE,
  digits = 3,
  nsmall = digits,
  file = NULL,
  check = TRUE,
  zero = ifelse(std, FALSE, TRUE),
  modify.se = NULL,
  modify.head = NULL,
  line = TRUE,
  bold = 0,
  ...
)
```
Arguments

model.list  A single model or a list of (various types of) models. Most types of regression models are supported!
std          Standardized coefficients? Default is FALSE. Only applicable to linear models and linear mixed models. Not applicable to generalized linear (mixed) models.
digits, nsmall  Number of decimal places of output. Default is 3.
file          File name of MS Word (.doc).
check         If there is only one model in model.list, it checks for multicollinearity using performance::check_collinearity(). You may turn it off by setting check=FALSE.
zero          Display "0" before "."? Default is TRUE.
modify.se     Replace standard errors. Useful if you need to replace raw SEs with robust SEs. New SEs should be provided as a list of numeric vectors. See usage in texreg::screenreg().
modify.head   Replace model names.
line          Lines look like true line (TRUE) or === ---=== (FALSE). Only relevant to R Console output.
bold          The p-value threshold below which the coefficients will be formatted in bold.
...           Other arguments passed to texreg::screenreg() or texreg::htmlreg().

Value

Invisibly return the output (character string).

See Also

print_table (print simple table)
GLM_summary
HLM_summary
med_summary
lavaan_summary
PROCESS

Examples

## Not run:

### Example 1: Linear Model ###
lm1 = lm(Temp ~ Month + Day, data=airquality)
lm2 = lm(Temp ~ Month + Day + Wind + Solar.R, data=airquality)
model_summary(lm1)
model_summary(lm2)
model_summary(list(lm1, lm2))
model_summary(list(lm1, lm2), std=TRUE, digits=2)
model_summary(list(lm1, lm2), file="OLS Models.doc")
### Example 2: Generalized Linear Model ###

```r
glm1 = glm(case ~ age + parity, 
            data=infert, family=binomial)
glm2 = glm(case ~ age + parity + education + spontaneous + induced, 
            data=infert, family=binomial)
model_summary(list(glm1, glm2))  # "std" is not applicable to glm
model_summary(list(glm1, glm2), file="GLM Models.doc")
unlink("GLM Models.doc")  # delete file for code check
```

### Example 3: Linear Mixed Model ###

```r
library(lmerTest)

hlm1 = lmer(Reaction ~ (1 | Subject), data=sleepstudy)
hlm2 = lmer(Reaction ~ Days + (1 | Subject), data=sleepstudy)
hlm3 = lmer(Reaction ~ Days + (Days | Subject), data=sleepstudy)
model_summary(list(hlm1, hlm2, hlm3))
model_summary(list(hlm1, hlm2, hlm3), std=TRUE)
model_summary(list(hlm1, hlm2, hlm3), file="HLM Models.doc")
unlink("HLM Models.doc")  # delete file for code check
```

### Example 4: Generalized Linear Mixed Model ###

```r
library(lmerTest)
data.glmm = MASS::bacteria
glmm1 = glmer(y ~ trt + week + (1 | ID), data=data.glmm, family=binomial)
glmm2 = glmer(y ~ trt + week + hilo + (1 | ID), data=data.glmm, family=binomial)
model_summary(list(glmm1, glmm2))  # "std" is not applicable to glmm
model_summary(list(glmm1, glmm2), file="GLMM Models.doc")
unlink("GLMM Models.doc")  # delete file for code check
```

### Example 5: Multinomial Logistic Model ###

```r
library(nnet)
d = airquality
d$Month = as.factor(d$Month)  # Factor levels: 5, 6, 7, 8, 9
mn1 = multinom(Month ~ Temp, data=d, Hess=TRUE)
mn2 = multinom(Month ~ Temp + Wind + Ozone, data=d, Hess=TRUE)
model_summary(mn1)
model_summary(mn2)
model_summary(mn2, file="Multinomial Logistic Model.doc")
unlink("Multinomial Logistic Model.doc")  # delete file for code check
```

## End(Not run)

---

**Description**

*Compute p value.*
Usage

\[
p(z = \text{NULL},
  t = \text{NULL},
  f = \text{NULL},
  r = \text{NULL},
  \chi^2 = \text{NULL},
  n = \text{NULL},
  df = \text{NULL},
  df1 = \text{NULL},
  df2 = \text{NULL},
  \text{digits} = 2,
  \text{nsmall} = \text{digits}
)
\]

\[p.z(z)\]

\[p.t(t, df)\]

\[p.f(f, df1, df2)\]

\[p.r(r, n)\]

\[p.chi2(chi2, df)\]

Arguments

- \(z, t, f, r, \chi^2\) \(z, t, F, r, \chi^2\) value.
- \(n, df, df1, df2\) Sample size or degree of freedom.
- \(\text{digits, nsmall}\) Number of decimal places of output. Default is 2.

Value

\(p\) value statistics.

Functions

- \(p.z\): Two-tailed \(p\) value of \(z\).
- \(p.t\): Two-tailed \(p\) value of \(t\).
- \(p.f\): One-tailed \(p\) value of \(F\). (Note: \(F\) test is one-tailed only.)
- \(p.r\): Two-tailed \(p\) value of \(r\).
- \(p.chi2\): One-tailed \(p\) value of \(\chi^2\). (Note: \(\chi^2\) test is one-tailed only.)

Examples

\[p.z(1.96)\]

\[p.t(2, 100)\]
pkg_depend

Check dependencies of R packages.

Description
Check dependencies of R packages.

Usage
pkg_depend(pkgs, excludes = NULL)

Arguments
pkgs Package(s).
excludes [optional] Package(s) and their dependencies excluded from the dependencies of pkgs. Useful if you want to see the unique dependencies of pkgs.

Value
A character vector of package names.

See Also
pkg_install_suggested

pkg_install_suggested
Install suggested R packages.

Description
Install suggested R packages.

Usage
pkg_install_suggested(by)
Arguments

by 
Suggested by which package?

Value

No return value.

See Also

pkg_depend

Examples

```r
## Not run:

pkg_install_suggested()  # install all packages suggested by me

## End(Not run)
```

---

**Print**

*Print strings with rich formats and colors.*

Description

Be frustrated with `print()` and `cat()`? Try `Print()`! Run examples to see what it can do.

Usage

`Print(...)`

`Glue(...)`

Arguments

... 
Character strings enclosed by "{}" will be evaluated as R code.
Character strings enclosed by "<<>>" will be printed as formatted and colored text.
Long strings are broken by line and concatenated together.
Leading whitespace and blank lines from the first and last lines are automatically trimmed.
Details

Possible formats/colors that can be used in "<< >>>" include:
(1) bold, italic, underline, reset, blurred, inverse, hidden, strikethrough;
(2) black, white, silver, red, green, blue, yellow, cyan, magenta;
(3) bgBlack, bgWhite, bgRed, bgGreen, bgBlue, bgYellow, bgCyan, bgMagenta.
See more details in glue::glue() and glue::glue_col().

Value

Formatted text.

Functions

- Print: Paste and print strings.
- Glue: Paste strings.

Examples

name = "Bruce"
Print("My name is <<underline <<bold {name}>>>>.
<<bold <<blue Pi = {pi:.15}.>>>>
<<italic <<green 1 + 1 = {1 + 1}.>>>>
sqrt({x}) = <<red {sqrt(x):.3}>>", x=10)

print_table

Print a three-line table (to R Console and Microsoft Word).

Description

This basic function prints any data frame as a three-line table to either R Console or Microsoft Word (.doc). It has been used in many other functions of bruceR (see below).

Usage

print_table(
  x,
  digits = 3,
  nsmalls = digits,
  nspaces = 1,
  row.names = TRUE,
  col.names = TRUE,
  title = "",
  note = "",
  append = "",
  line = TRUE,
)
print_table

```r
file = NULL,
file.align.head = "auto",
file.align.text = "auto"
)
```

**Arguments**

- **x**: Matrix, data.frame (or data.table), or any model object (e.g., `lm`, `glm`, `lmer`, `glmer`, ...).
- **digits, nsmalls**: Numeric vector specifying the number of decimal places of output. Default is 3.
- **nspaces**: Number of whitespaces between columns. Default is 1.
- **row.names, col.names**: Print row/column names. Default is TRUE (column names are always printed). To modify the names, you can use a character vector with the same length as the raw names.
- **title**: Title text, which will be inserted in `<p></p>` (HTML code).
- **note**: Note text, which will be inserted in `<p></p>` (HTML code).
- **append**: Other contents, which will be appended in the end (HTML code).
- **line**: Lines looks like true line (TRUE) or `===` `=====` (FALSE).
- **file**: File name of MS Word (.doc).
- **file.align.head, file.align.text**: Alignment of table head or table text: "left", "right", "center". Either one value of them OR a character vector of mixed values with the same length as the table columns. Default alignment (if set as "auto"): left, right, right, ..., right.

**Value**

Invisibly return a list of data frame and HTML code.

**See Also**

These functions have implemented MS Word file output using this function:

- Describe
- Freq
- Corr
- EFA / PCA
- CFA
- TTEST
- MANOVA
- model_summary
- med_summary
- lavaan_summary
- PROCESS
- granger_test
- granger_causality
**Examples**

```r
print_table(data.frame(x=1))

print_table(airquality, file="airquality.doc")
unlink("airquality.doc")  # delete file for code check

model = lm(Temp ~ Month + Day + Wind + Solar.R, data=airquality)
print_table(model)
print_table(model, file="model.doc")
unlink("model.doc")  # delete file for code check
```

---

**Description**

To perform mediation, moderation, and conditional process (moderated mediation) analyses, people may use software like Mplus, SPSS "PROCESS" macro, and SPSS "MLmed" macro. Some R packages can also perform such analyses separately and in a complex way, including R package "mediation", R package "interactions", and R package "lavaan". Some other R packages or scripts/modules have been further developed to improve the convenience, including jamovi module "jAMM" (by Marcello Gallucci, based on the lavaan package), R package "processR" (by Keon-Woong Moon, not official, also based on the lavaan package), and R script file "process.R" (the official PROCESS R code by Andrew F. Hayes, but it is not yet an R package and has some bugs and limitations).

Here, the `bruceR::PROCESS()` function provides an alternative to performing mediation/moderation analyses in R. This function supports a total of 24 kinds of SPSS PROCESS models (Hayes, 2018) and also supports multilevel mediation/moderation analyses. Overall, it supports the most frequently used types of mediation, moderation, moderated moderation (3-way interaction), and moderated mediation (conditional indirect effect) analyses for (generalized) linear or linear mixed models.

Specifically, the `bruceR::PROCESS()` function fits regression models based on the data, variable names, and a few other arguments that users input (with no need to specify the PROCESS model number and no need to manually mean-center the variables). The function can automatically judge the model number/type and also conduct grand-mean centering before model building (using the `bruceR::grand_mean_center()` function).

This automatic grand-mean centering can be turned off by setting center=FALSE.

Note that this automatic grand-mean centering (1) makes the results of main effects accurate for interpretation; (2) does not change any results of model fit (it only affects the interpretation of main effects); (3) is only conducted in "PART 1" (for an accurate estimate of main effects) but not in "PART 2" because it is more intuitive and interpretable to use the raw values of variables for the simple-slope tests in "PART 2"; (4) is not optional to users because mean-centering should always be done when there is an interaction; (5) is not conflicted with group-mean centering because after group-mean centering the grand mean of a variable will also be 0, such that the automatic grand-mean centering (with mean = 0) will not change any values of the variable.
If you need to do group-mean centering, please do this before using PROCESS. `bruceR::group_mean_center()` is a useful function of group-mean centering. Remember that the automatic grand-mean centering in PROCESS never affects the values of a group-mean centered variable, which already has a grand mean of 0.

The `bruceR::PROCESS()` function uses:

1. the `interactions::sim_slopes()` function to estimate simple slopes (and conditional direct effects) in moderation, moderated moderation, and moderated mediation models (PROCESS Models 1, 2, 3, 5, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, 58, 59, 72, 73, 75, 76).
2. the `mediation::mediate()` function to estimate (conditional) indirect effects in (moderated) mediation models (PROCESS Models 4, 5, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, 58, 59, 72, 73, 75, 76).
3. the `lavaan::sem()` function to perform serial multiple mediation analysis (PROCESS Model 6).

If you use this function in your research and report its results in your paper, please cite not only `bruceR` but also the other R packages it uses internally (`mediation`, `interactions`, and/or `lavaan`).

Two parts of results are printed:

PART 1. Regression model summary (using `bruceR::model_summary()` to summarize the models)

PART 2. Mediation/moderation effect estimates (using one or a combination of the above packages and functions to estimate the effects)

To organize the PART 2 output, the results of Simple Slopes are titled in green, whereas the results of Indirect Path are titled in blue.

Disclaimer: Although this function is named after PROCESS, Andrew F. Hayes has no role in its design, and its development is independent from the official SPSS PROCESS macro and "process.R" script. Any error or limitation should be attributed to the three R packages/functions that `bruceR::PROCESS()` uses internally. Moreover, as mediation analyses include random processes (i.e., bootstrap resampling or Monte Carlo simulation), the results of mediation analyses are unlikely to be exactly the same across different software (even if you set the same random seed in different software).

Usage

```r
PROCESS(
  data,
  y = "",
  x = "",
  meds = c(),
  mods = c(),
  covs = c(),
  clusters = c(),
  hlm.re.m = "",
  hlm.re.y = "",
  hlm.type = c("1-1-1", "2-1-1", "2-2-1"),
  med.type = c("parallel", "serial"),
  mod.type = c("2-way", "3-way"),
  mod.path = c("x-y", "x-m", "m-y", "all"),
)```
cov.path = c("y", "m", "both"),
mod1.val = NULL,
mod2.val = NULL,
c1 = c("boot", "bc.boot", "bca.boot", "mcmc"),
nsim = 100,
seed = NULL,
center = TRUE,
std = FALSE,
digits = 3,
nsmall = digits,
file = NULL
)

Arguments

data Data frame.
y, x Variable name of outcome (Y) and predictor (X).
It supports both continuous (numeric) and dichotomous (factor) variables.
meds Variable name(s) of mediator(s) (M). Use c() to combine multiple mediators.
It supports both continuous (numeric) and dichotomous (factor) variables.
It allows an infinite number of mediators in parallel or 2~4 mediators in serial.
* Order matters when med.type="serial" (PROCESS Model 6: serial mediation).
mods Variable name(s) of 0~2 moderator(s) (W). Use c() to combine multiple moderators.
It supports all types of variables: continuous (numeric), dichotomous (factor),
and multicategorical (factor).
* Order matters when mod.type="3-way" (PROCESS Models 3, 5.3, 11, 12,
18, 19, 72, and 73).
** Do not set this argument when med.type="serial" (PROCESS Model 6).
covs Variable name(s) of covariate(s) (i.e., control variables). Use c() to combine multiple covariates. It supports all types of (and an infinite number of) variables.
clusters HLM (multilevel) cluster(s): e.g., "School", c("Prov","City"), c("Sub","Item").
hlm.re.m, hlm.re.y HLM (multilevel) random effect term of M model and Y model. By default, it converts clusters to lme4 syntax of random intercepts: e.g., "(1 | School)" or "(1 | Sub) + (1 | Item)".
You may specify these arguments to include more complex terms: e.g., random slopes "(X | School)" , or 3-level random effects "(1 | Prov/City)".
hlm.type HLM (multilevel) mediation type (levels of "X-M-Y"): "1-1-1" (default), "2-1-1" (indeed the same as "1-1-1" in a mixed model), or "2-2-1" (currently not fully supported, as limited by the mediation package). In most cases, no need to set this argument.
med.type Type of mediator: "parallel" (default) or "serial" (only relevant to PROCESS Model 6). Partial matches of "p" or "s" also work. In most cases, no need to set this argument.
**mod.type**
Type of moderator: "2-way" (default) or "3-way" (relevant to PROCESS Models 3, 5.3, 11, 12, 18, 19, 72, and 73). Partial matches of "2" or "3" also work.

**mod.path**
Which path(s) do the moderator(s) influence? "x-y", "x-m", "m-y", or any combination of them (use c( ) to combine), or "all" (i.e., all of them). No default value.

**cov.path**
Which path(s) do the control variable(s) influence? "y", "m", or "both" (default).

**mod1.val, mod2.val**
By default (NULL), it uses Mean +/- SD of a continuous moderator (numeric) or all levels of a dichotomous/multicategorical moderator (factor) to perform simple slope analyses and/or conditional mediation analyses. You may manually specify a vector of certain values: e.g., mod1.val=c(1, 3, 5) or mod1.val=c("A", "B", "C").

**ci**
Method for estimating the standard error (SE) and 95% confidence interval (CI) of indirect effect(s). Default is "boot" for (generalized) linear models or "mcmc" for (generalized) linear mixed models (i.e., multilevel models).
- "boot" Percentile Bootstrap
- "bc.boot" Bias-Corrected Percentile Bootstrap
- "bca.boot" Bias-Corrected and Accelerated (BCa) Percentile Bootstrap
- "mcmc" Markov Chain Monte Carlo (Quasi-Bayesian)

* Note that these methods never apply to the estimates of simple slopes. You should not report the 95% CIs of simple slopes as Bootstrap or Monte Carlo CIs, because they are just standard CIs without any resampling method.

**nsim**
Number of simulation samples (bootstrap resampling or Monte Carlo simulation) for estimating SE and 95% CI. Default is 100 for running examples faster. In formal analyses, however, nsim=1000 (or larger) is strongly suggested!

**seed**
Random seed for obtaining reproducible results. Default is NULL. You may set to any number you prefer (e.g., seed=1234, just an uncountable number).

* Note that all mediation models include random processes (i.e., bootstrap resampling or Monte Carlo simulation). To get exactly the same results between runs, you need to set a random seed. However, even if you set the same seed number, it is unlikely to get exactly the same results across different R packages (e.g., lavaan vs. mediation) and software (e.g., SPSS, Mplus, R, jamovi).

**center**
Centering numeric (continuous) predictors? Default is TRUE (suggested).

**std**
Standardizing variables to get standardized coefficients? Default is FALSE. If TRUE, it will standardize all numeric (continuous) variables before building regression models. However, it is not suggested to set std=TRUE for generalized linear (mixed) models.

**digits, nsmall**
Number of decimal places of output. Default is 3.

**file**
File name of MS Word (.doc). Currently, only regression model summary can be saved.

**Details**
For more details and illustrations, see PROCESS-bruceR-SPSS (PDF and Markdown files).
Value

Invisibly return a list of results:

- **process.id**  PROCESS model number.
- **process.type**  PROCESS model type.
- **model.m**  "Mediator" (M) models (a list of multiple models).
- **model.y**  "Outcome" (Y) model.
- **results**  Effect estimates and other results (unnamed list object).

References


See Also

- `lavaan_summary`
- `model_summary`
- `med_summary`

Examples

```r
## Not run:
### NOTE ####
## In the following examples, I set nsim=100 to save time.
## In formal analyses, nsim=1000 (or larger) is suggested!

### Demo Data ####
data = mediation::student %>%
dplyr::select(SCH_ID, free, smorale, pared, income,
              gender, work, attachment, fight, late, score)
names(data)[2:3] = c("SCH_free", "SCH_morale")
names(data)[4:7] = c("parent_edu", "family_inc", "gender", "partjob")
data$gender01 = 1 - data$gender  # 0 = female, 1 = male
# dichotomous X: as.factor()
data$gender = factor(data$gender01, levels=0:1, labels=c("Female", "Male"))
# dichotomous Y: as.factor()
data$pass = as.factor(ifelse(data$score>=50, 1, 0))

### Descriptive Statistics and Correlation Analyses ####
Freq(data$gender)
Freq(data$pass)
Describe(data)  # file="xxx.doc"
Corr(data[,4:11])  # file="xxx.doc"
```
### PROCESS Analyses ###

**Model 1**
- `PROCESS(data, y="score", x="late", mods="gender")` # continuous Y
- `PROCESS(data, y="pass", x="late", mods="gender")` # dichotomous Y

# (multilevel moderation)
- `PROCESS(data, y="score", x="late", mods="gender", clusters="SCH_ID")` # continuous Y (LMM)
- `PROCESS(data, y="pass", x="late", mods="gender", clusters="SCH_ID")` # dichotomous Y (GLMM)

# (Johnson-Neyman (J-N) interval and plot)
- `P$results[[1]]$jn[[1]]` # Johnson-Neyman interval
- `P$results[[1]]$jn[[1]]$plot` # Johnson-Neyman plot (ggplot object)
- `GLM_summary(P$model.y)` # detailed results of regression

# (allows multicategorical moderator)
- `d = airquality`
- `d$Month = as.factor(d$Month)` # moderator: factor with levels "5"~"9"
- `PROCESS(d, y="Temp", x="Solar.R", mods="Month")`

**Model 2**
- `PROCESS(data, y="score", x="late", mods=c("gender", "family_inc"), mod.type="2-way")` # or omit "mod.type", default is "2-way"

**Model 3**
- `PROCESS(data, y="score", x="late", mods=c("gender", "family_inc"), mod.type="3-way")`
- `PROCESS(data, y="pass", x="gender", mods=c("late", "family_inc"), mod1.val=c(1, 3, 5), mod2.val=seq(1, 15, 2), mod.type="3-way")`

**Model 4**
- `PROCESS(data, y="score", x="parent_edu", meds="family_inc", covs="gender", ci="boot", nsim=100, seed=1)` # (allows an infinite number of multiple mediators in parallel)
- `PROCESS(data, y="score", x="parent_edu", meds=c("family_inc", "late"), covs=c("gender", "partjob"), ci="boot", nsim=100, seed=1)` # (multilevel mediation)
- `PROCESS(data, y="score", x="SCH_free", meds="late", clusters="SCH_ID",`
RECODE

R code example:

```r
## Model 6 ##
PROCESS(data, y="score", x="parent_edu",
meds=c("family_inc", "late"),
covs=c("gender", "partjob"),
med.type="serial",
ci="mcmc", nsim=100, seed=1)

## Model 8 ##
PROCESS(data, y="score", x="fight",
meds="late",
mods="gender",
mod.path=c("x-m", "x-y"),
ci="boot", nsim=100, seed=1)
```

## For more examples and details, see the "note" subfolder at:
## https://github.com/psychbruce/bruceR/tree/main/note
## End(Not run)

---

**RECODE**

*Recode a variable.*

**Description**

A wrapper of `car::recode()`.

**Usage**

```r
RECODE(var, recodes)
```

**Arguments**

- `var` Variable (numeric, character, or factor).
- `recodes` A character string defining the rule of recoding. e.g., "lo:1=0; c(2,3)=1; 4=2; 5:hi=3; else=999"

**Value**

A vector of recoded variable.

**Examples**

```r
d = data.table(var=c(NA, 0, 1, 2, 3, 4, 5, 6))
d[, `:=`
  var.new = RECODE(var, "lo:1=0; c(2,3)=1; 4=2; 5:hi=3; else=999")
]
d
```
**regress**

*Regression analysis.*

**Description**

NOTE: *model_summary* is preferred.

**Usage**

```r
regress(
  formula,
  data,
  family = NULL,
  digits = 3,
  nsmall = digits,
  robust = FALSE,
  cluster = NULL,
  test.rand = FALSE
)
```

**Arguments**

- `formula` Model formula.
- `data` Data frame.
- `family` [Optional] The same as in `glm` and `glmer` (e.g., `family=binomial` fits a logistic regression model).
- `digits` Number of decimal places of output. Default is 3.
- `nsmall` Number of decimal places of output. Default is 3.
- `robust` [Only for `lm` and `glm`] FALSE (default), TRUE (then the default is "HC1"), "HC0", "HC1", "HC2", "HC3", "HC4", "HC4m", or "HC5". It will add a table with heteroskedasticity-robust standard errors (aka. Huber-White standard errors). For details, see `?sandwich::vcovHC` and `?jtools::summ.lm`.
  
  *** "HC1" is the default of Stata, whereas "HC3" is the default suggested by the sandwich package.

- `cluster` [Only for `lm` and `glm`] Cluster-robust standard errors are computed if cluster is set to the name of the input data’s cluster variable or is a vector of clusters.
- `test.rand` [Only for `lmer` and `glmer`] TRUE or FALSE (default). Test random effects (i.e., variance components) by using the likelihood-ratio test (LRT), which is asymptotically chi-square distributed. For large datasets, it is much time-consuming.

**Value**

No return value.
rep_char

Repeat a character string for many times and paste them up.

Description

Repeat a character string for many times and paste them up.

Usage

rep_char(char, rep.times)

Arguments

char Character string.
rep.times Times for repeat.
Value
Character string.

Examples
rep_char("a", 5)

RESCALE

Rescale a variable (e.g., from 5-point to 7-point).

Description
Rescale a variable (e.g., from 5-point to 7-point).

Usage
RESCALE(var, from = range(var, na.rm = T), to)

Arguments
var Variable (numeric).
from Numeric vector, the range of old scale (e.g., 1:5). If not defined, it will compute the range of var.
to Numeric vector, the range of new scale (e.g., 1:7).

Value
A vector of rescaled variable.

Examples
d = data.table(var=rep(1:5, 2))
d[, `:=`(var1 = RESCALE(var, to=1:7),
    var2 = RESCALE(var, from=1:5, to=1:7))]
d # var1 is equal to var2
**RGB**

A simple extension of `rgb()`.

**Description**

A simple extension of `rgb()`.

**Usage**

```r
RGB(r, g, b, alpha)
```

**Arguments**

- `r, g, b` Red, Green, Blue: 0–255.
- `alpha` Color transparency (opacity): 0–1. If not specified, an opaque color will be generated.

**Value**

"#rrggbb" or "#rrggbbaa".

**Examples**

```r
RGB(255, 0, 0) # red: "#FF0000"
RGB(255, 0, 0, 0.8) # red with 80% opacity: "#FF0000CC"
```

---

**Run**

Run code parsed from text.

**Description**

Run code parsed from text.

**Usage**

```r
Run(..., silent = FALSE)
```

**Arguments**

- `...` Character string(s) to run. You can use "( )" to insert any R object in the environment.
- `silent` Suppress error/warning messages. Default is FALSE.
Value
Invisibly return the running expression(s).

Examples
Run("a=1", "b=2")
Run("print({a+b})")

scaler

Min-max scaling (min-max normalization).

Description
This function resembles RESCALE() and it is just equivalent to RESCALE(var, to=0:1).

Usage
scaler(v, min = 0, max = 1)

Arguments
- v Variable (numeric vector).
- min Minimum value (default is 0).
- max Maximum value (default is 1).

Value
A vector of rescaled variable.

Examples
scaler(1:5)
# the same: RESCALE(1:5, to=0:1)
set.wd

Set working directory to the path of currently opened file.

Description

Set working directory to the path of currently opened file (usually an R script). You can use this function in both .R/.Rmd files and R Console. RStudio (version >= 1.2) is required for running this function.

Usage

```r
set.wd(path = NULL, ask = FALSE)
set_wd(path = NULL, ask = FALSE)
```

Arguments

- **path** NULL (default) or a specific path. Default is to extract the path of the currently opened file (usually .R or .Rmd) using the `rstudioapi::getSourceEditorContext` function.
- **ask** TRUE or FALSE (default). If TRUE, you can select a folder with the prompt of a dialog.

Value

Invisibly return the path.

Functions

- `set.wd`: Main function
- `set_wd`: The alias of `set.wd` (the same)

See Also

- `setwd`

Examples

```r
## Not run:

# RStudio (version >= 1.2) is required for running this function.
set.wd()  # set working directory to the path of the currently opened file
set.wd("~")  # set working directory to the home path
set.wd("../")  # set working directory to the parent path
set.wd(ask=TRUE)  # select a folder with the prompt of a dialog

## End(Not run)
```
show_colors

Show colors.

Description

Show colors.

Usage

```r
show_colors(colors = see::social_colors())
```

Arguments

colors

Color names.

e.g.,

- "red" (R base color names)
- "#FF0000" (hex color names)
- see::social_colors()
- `viridis::viridis_pal()`(10)
- `RCOLORBrewer::brewer.pal(name="Set1",n=9)`
- `RCOLORBrewer::brewer.pal(name="Set2",n=8)`
- `RCOLORBrewer::brewer.pal(name="Spectral",n=11)`

Value

A gg object.

Examples

```r
show_colors() # default is to show see::social_colors()
show_colors("blue") # blue
show_colors("#0000FF") # blue (hex name)
show_colors(RGB(0, 0, 255)) # blue (RGB)
show_colors(see::pizza_colors()) # a specific palette
```
theme_bruce

A nice ggplot2 theme that enables Markdown/HTML rich text.

Description

A nice ggplot2 theme for scientific publication. It uses \texttt{ggtext::element_markdown()} to render Markdown/HTML formatted rich text. You can use a combination of Markdown and/or HTML syntax (e.g., "y = x^2") in plot text or title, and this function draws text elements with rich text format.

For more usage, see:

- \texttt{ggtext::geom_richtext()}
- \texttt{ggtext::geom_textbox()}
- \texttt{ggtext::element_markdown()}
- \texttt{ggtext::element_textbox()}

Usage

```r
theme_bruce(
  markdown = FALSE,
  base.size = 12,
  line.size = 0.5,
  border = "black",
  bg = "white",
  panel.bg = "white",
  tag = "bold",
  plot.title = "bold",
  axis.title = "plain",
  title.pos = 0.5,
  subtitle.pos = 0.5,
  caption.pos = 1,
  font = NULL,
  grid.x = "",
  grid.y = "",
  line.x = TRUE,
  line.y = TRUE,
  tick.x = TRUE,
  tick.y = TRUE
)
```

Arguments

- \texttt{markdown} Use \texttt{element_markdown()} instead of \texttt{element_text()}. Default is \texttt{FALSE}. If set to \texttt{TRUE}, then you should also use \texttt{element_markdown()} in \texttt{theme()} (if any).
- \texttt{base.size} Basic font size. Default is 12.
### line.size

Line width. Default is 0.5.

### border

TRUE, FALSE, or "black" (default).

### bg

Background color of whole plot. Default is "white". You can use any colors or choose from some pre-set color palettes: "stata","stata.grey","solar","wsj","light","dust". To see these colors, you can type:

```
ggthemr::colour_plot(c(stata="#EAF2F3",stata.grey="#E8E8E8",solar="#FDF6E3",wsj="#F8F2E4",light="#F6F1EB",dust="#FAF7F2"))
```

### panel.bg

Background color of panel. Default is "white".

### tag

Font face of tag. Choose from "plain", "italic", "bold", "bold.italic".

### plot.title

Font face of title. Choose from "plain", "italic", "bold", "bold.italic".

### axis.title

Font face of axis text. Choose from "plain", "italic", "bold", "bold.italic".

### title.pos

Title position (0~1).

### subtitle.pos

Subtitle position (0~1).

### caption.pos

Caption position (0~1).

### font

Text font. Only applicable to Windows system.

### grid.x

FALSE, "" (default), or a color (e.g., "grey90") to set the color of panel grid (x).

### grid.y

FALSE, "" (default), or a color (e.g., "grey90") to set the color of panel grid (y).

### line.x

Draw the x-axis line. Default is TRUE.

### line.y

Draw the y-axis line. Default is TRUE.

### tick.x

Draw the x-axis ticks. Default is TRUE.

### tick.y

Draw the y-axis ticks. Default is TRUE.

### Value

A theme object that should be used for ggplot2.

### Examples

#### Example 1 (bivariate correlation)

```r
D = as.data.table(psych::bfi)
D[, z := (E = MEAN(D, "E", 1:5, rev=c(1,2), likert=1:6),
       O = MEAN(D, "O", 1:5, rev=c(2,5), likert=1:6))]

ggplot(data=D, aes(x=E, y=O)) +
   geom_point(alpha=0.1) +
   geom_smooth(method="loess") +
   labs(x="Extraversion<sub>Big 5</sub>",
        y="Openness<sub>Big 5</sub>") +
   theme_bruce(markdown=TRUE)
```

#### Example 2 (2x2 ANOVA)

```r
D = data.frame(X1 = factor(rep(1:3, each=2)),
               X2 = factor(rep(1:2, 3)),
               Y.mean = c(5, 3, 2, 7, 3, 6),
               Y.se = rep(c(0.1, 0.2, 0.1), each=2))

ggplot(data=D, aes(x=X1, y=Y.mean, fill=X2)) +
   geom_bar(position="dodge", stat="identity", width=0.6, show.legend=FALSE) +
```
TTEST

One-sample, independent-samples, and paired-samples t-test.

Description

One-sample, independent-samples, and paired-samples t-test, with both Frequentist and Bayesian approaches. The output includes descriptives, t statistics, mean difference with 95% CI, Cohen’s d with 95% CI, and Bayes factor (BF10). It also tests the assumption of homogeneity of variance and allows users to determine whether variances are equal or not.

Users can simultaneously test multiple dependent and/or independent variables. The results of one pair of Y-X would be summarized in one row in the output. Key results can be saved in APA format to MS Word.

Usage

TTEST(
  data,
  y,
  x = NULL,
  paired = FALSE,
  var.equal = TRUE,
  mean.diff = TRUE,
  test.value = 0,
  test.sided = c("="", "<", ">"),
  factor.rev = TRUE,
  bayes.prior = "medium",
  digits = 2,
  nsmall = digits,
  file = NULL
)

Arguments

data  Data frame (wide-format only, i.e., one case in one row).
y  Dependent variable(s). Multiple variables should be included in a character vector c().

For paired-samples t-test, the number of variables should be 2, 4, 6, etc.
Independent variable(s). Multiple variables should be included in a character vector `c()`. Only necessary for independent-samples `t`-test.

`paired` For paired-samples `t`-test, set it to `TRUE`. Default is `FALSE`.

`var.equal` If Levene’s test indicates a violation of the homogeneity of variance, then you should better set this argument to `FALSE`. Default is `TRUE`.

`mean.diff` Whether to display results of mean difference and its 95% CI. Default is `TRUE`.

`test.value` The true value of the mean (or difference in means for a two-samples test). Default is `0`.

`test.sided` Any of `"="` (two-sided, the default), `"<"` (one-sided), or `">"` (one-sided).

`factor.rev` Whether to reverse the levels of factor (X) such that the test compares higher vs. lower level. Default is `TRUE`.

`bayes.prior` Prior scale in Bayesian `t`-test. Default is `0.707`. See details in `BayesFactor::ttestBF()`.

`digits`, `nsmall` Number of decimal places of output. Default is `2`.

`file` File name of MS Word (.doc).

**Details**

Note that the point estimate of Cohen's `d` is computed using the common method "Cohen's `d` = mean difference / (pooled) standard deviation", which is consistent with results from other R packages (e.g., `effectsize`) and software (e.g., `jamovi`). The 95% CI of Cohen’s `d` is estimated based on the 95% CI of mean difference (i.e., also divided by the pooled standard deviation).

However, different packages and software diverge greatly on the estimate of the 95% CI of Cohen’s `d`. R packages such as `psych` and `effectsize`, R software `jamovi`, and several online statistical tools for estimating effect sizes indeed produce surprisingly inconsistent results on the 95% CI of Cohen’s `d`.

See an illustration of this issue in the section "Examples".

**See Also**

`MANOVA`, `EMMEANS`

**Examples**

```r
## Demo data ##
d1 = between.3
d1$Y1 = d1$SCORE  # shorter name for convenience
d1$Y2 = rnorm(32)  # random variable
d1$B = factor(d1$B, levels=1:2, labels=c("Low", "High"))
d1$C = factor(d1$C, levels=1:2, labels=c("M", "F"))
d2 = within.1

## One-sample t-test ##
TTEST(d1, "SCORE")
TTEST(d1, "SCORE", test.value=5)

## Independent-samples t-test ##
```
TTEST(d1, "SCORE", x="A")
TTEST(d1, "SCORE", x="A", var.equal=FALSE)
TTEST(d1, y="Y1", x=c("A", "B", "C"))
TTEST(d1, y=c("Y1", "Y2"), x=c("A", "B", "C"),
    mean.diff=FALSE, # remove to save space
    file="t-result.doc")
unlink("t-result.doc") # delete file for code check

## Paired-samples t-test ##
TTEST(d2, y=c("A1", "A2"), paired=TRUE)
TTEST(d2, y=c("A1", "A2", "A3", "A4"), paired=TRUE)

## Not run:

## Illustration for the issue stated in "Details"
# Inconsistency in the 95% CI of Cohen's d between R packages:
# In this example, the true point estimate of Cohen's d = 3.00
# and its 95% CI should be equal to 95% CI of mean difference.
data = data.frame(X=rep(1:2, each=3), Y=1:6)
data # simple demo data
TTEST(data, y="Y", x="X")
# d = 3.00 [0.73, 5.27] (estimated based on 95% CI of mean difference)
MANOVA(data, dv="Y", between="X") %>%
    EMMEANS("X")
# d = 3.00 [0.73, 5.27] (the same as TTEST)
psych::cohen.d(x=data, group="X")
# d = 3.67 [0.04, 7.35] (strange)
psych::d.ci(d=3.00, n1=3, n2=3)
# d = 3.00 [-0.15, 6.12] (significance inconsistent with t-test)
# jamovi uses psych::d.ci() to compute 95% CI
# so its results are also: 3.00 [-0.15, 6.12]
effectsize::cohens_d(Y ~ rev(X), data=data)
# d = 3.00 [0.38, 5.50] (using the noncentrality parameter method)
effectsize::t_to_d(t=t.test(Y ~ rev(X), data=data, var.equal=TRUE)$statistic,
    df_error=4)
# d = 3.67 [0.47, 6.74] (merely an approximate estimate, often overestimated)
# see ?effectsize::t_to_d

# https://www.psychometrica.de/effect_size.html
# d = 3.00 [0.67, 5.33] (slightly different from TTEST)
# https://www.campbellcollaboration.org/escalc/
# d = 3.00 [0.67, 5.33] (slightly different from TTEST)
# Conclusion:
# TTEST() provides a reasonable estimate of Cohen's d and its 95% CI, 
# and effectsize::cohens_d() offers another method to compute the CI.

## End(Not run)

---

%allin%  

A simple extension of %in%.

## Description

A simple extension of %in%.

## Usage

```r
x %allin% vector
```

## Arguments

- `x`: Numeric or character vector.
- `vector`: Numeric or character vector.

## Value

TRUE or FALSE.

## See Also

%in%, %anyin%, %nonein%, %partin%

## Examples

```r
1:2 %allin% 1:3 # TRUE
3:4 %allin% 1:3 # FALSE
```
%anyin%  

A simple extension of %in%.

Description

A simple extension of %in%.

Usage

x %anyin% vector

Arguments

x Numeric or character vector.
vector Numeric or character vector.

Value

TRUE or FALSE.

See Also

%in%, %allin%, %nonein%, %partin%

Examples

3:4 %anyin% 1:3 # TRUE
4:5 %anyin% 1:3 # FALSE

%%COMPUTE%%  

Multivariate computation.

Description

Easily compute multivariate sum, mean, and other scores. Reverse scoring can also be easily implemented without saving extra variables. Alpha function uses a similar method to deal with reverse scoring.

Three options to specify variables:

1. var + items: use the common and unique parts of variable names.
2. vars: directly define a character vector of variables.
3. varrange: use the starting and stopping positions of variables.
Usage

COUNT(data, var = NULL, items = NULL, vars = NULL, varrange = NULL, value = NA)

MODE(data, var = NULL, items = NULL, vars = NULL, varrange = NULL)

SUM(
    data,
    var = NULL,
    items = NULL,
    vars = NULL,
    varrange = NULL,
    rev = NULL,
    likert = NULL,
    na.rm = TRUE
)

MEAN(
    data,
    var = NULL,
    items = NULL,
    vars = NULL,
    varrange = NULL,
    rev = NULL,
    likert = NULL,
    na.rm = TRUE
)

STD(
    data,
    var = NULL,
    items = NULL,
    vars = NULL,
    varrange = NULL,
    rev = NULL,
    likert = NULL,
    na.rm = TRUE
)

CONSEC(
    data,
    var = NULL,
    items = NULL,
    vars = NULL,
    varrange = NULL,
    values = 0:9
)
Arguments

data
  Data frame.

var
  [Option 1] The common part across the variables. e.g., "RSES"

items
  [Option 1] The unique part across the variables. e.g., 1:10

vars
  [Option 2] A character vector specifying the variables. e.g., c("X1", "X2", "X3", "X4", "X5")

varrange
  [Option 3] A character string specifying the positions ("starting:stopping") of variables. e.g., "A1:E5"

value
  [Only for COUNT] The value to be counted.

rev
  [Optional] Variables that need to be reversed. It can be (1) a character vector specifying the reverse-scoring variables (recommended), or (2) a numeric vector specifying the item number of reverse-scoring variables (not recommended).

likert
  [Optional] Range of likert scale (e.g., 1:5, c(1,5)). If not provided, it will be automatically estimated from the given data (BUT you should use this carefully).

na.rm
  Ignore missing values. Default is TRUE.

values
  [Only for CONSEC] Values to be counted as consecutive identical values. Default is all numbers (0:9).

Value

A vector of computed values.

Functions

- **COUNT**: Count a certain value across multiple variables.
- **MODE**: Compute mode across multiple variables.
- **SUM**: Compute sum across multiple variables.
- **MEAN**: Compute mean across multiple variables.
- **STD**: Compute standard deviation across multiple variables.
- **CONSEC**: Compute consecutive identical digits across multiple variables (especially useful in detecting careless responding).

Examples

```r
d = data.table(x1=1:5,
  x4=c(2,2,5,4,5),
  x3=c(3,2,NA,NA,5),
  x2=c(4,4,NA,2,5),
  x5=c(5,4,1,4,5))
d
# I deliberately set this order to show you
# the difference between "vars" and "varrange".

d[, `:=`(
  na = COUNT(d, "x", 1:5, value=NA),
  n.2 = COUNT(d, "x", 1:5, value=2),
```

```
sum = SUM(d, "x", 1:5),
m1 = MEAN(d, "x", 1:5),
m2 = MEAN(d, vars=c("x1", "x4")),
m3 = MEAN(d, varrange="x1:x2", rev="x2", likert=1:5),
cons1 = CONSEC(d, "x", 1:5),
cons2 = CONSEC(d, varrange="x1:x5")
}

d

data = as.data.table(psych::bfi)
data[, `:=`(  
  E = MEAN(d, "E", 1:5, rev=c(1,2), likert=1:6),  
  O = MEAN(d, "O", 1:5, rev=c(2,5), likert=1:6)
)]
data

%nonein% A simple extension of %in%.

Description
A simple extension of %in%.

Usage
x %nonein% vector

Arguments
x Numeric or character vector.
vector Numeric or character vector.

Value
TRUE or FALSE.

See Also
%in%,%allin%,%anyin%,%partin%

Examples
3:4 %nonein% 1:3 # FALSE
4:5 %nonein% 1:3 # TRUE
**%notin%**

The opposite of `%in%`.

**Description**

The opposite of `%in%`.

**Usage**

```
x %notin% vector
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Numeric or character vector.</td>
</tr>
<tr>
<td>vector</td>
<td>Numeric or character vector.</td>
</tr>
</tbody>
</table>

**Value**

A vector of TRUE or FALSE.

**See Also**

`%in%`

**Examples**

```
data = data.table(ID=1:10, X=sample(1:10, 10))
data
data[ID %notin% c(1, 3, 5, 7, 9)]
```

---

**%partin%**

A simple extension of `%in%`.

**Description**

A simple extension of `%in%`.

**Usage**

```
pattern %partin% vector
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pattern</td>
<td>Character string containing regular expressions to be matched.</td>
</tr>
<tr>
<td>vector</td>
<td>Character vector.</td>
</tr>
</tbody>
</table>
Value

TRUE or FALSE.

See Also

%in%, %allin%, %anyin%, %nonein%

Examples

"Bei" %partin% c("Beijing", "Shanghai") # TRUE
"bei" %partin% c("Beijing", "Shanghai") # FALSE
"[aeiou]ng" %partin% c("Beijing", "Shanghai") # TRUE

Description

Paste strings together. A wrapper of paste0(). Why %^%^? Because typing % and ^ is pretty easy by pressing Shift + 5 + 6 + 5.

Usage

x %^%^ y

Arguments

x, y Any objects, usually a numeric or character string or vector.

Value

A character string/vector of the pasted values.

Examples

"He" %^%^ "llo"
"X" %^%^ 1:10
"Q" %^%^ 1:5 %^%^ letters[1:5]
Index

%%COMPUTE%%, 75
%%, 4, 80
%allin%, 4, 74, 75, 78, 80
%anyin%, 4, 74, 75, 78, 80
%in%, 74, 75, 78–80
%nonein%, 4, 74, 75, 78, 80
%notin%, 4, 79
%partin%, 4, 74, 75, 78, 79

afex::aov_ez(), 19, 42–44
afex::contrast(), 18
afex::eff_size(), 19
afex::emmeans(), 18
afex::pairwise.t.test(), 72
afex::summary(), 19
dbase::load(), .37
dbase::readRDS(), .37
dbase::save(), .23
dbase::saveRDS(), .23
BayesFactor::ttestBF(), .72
between.1(bruceR-demodata), 6
between.2(bruceR-demodata), 6
between.3(bruceR-demodata), 6
bruceR(bruceR-package), 3
bruceR-demodata, 6
bruceR-package, 3
bruceR::grand_mean_center(), .55
bruceR::group_mean_center(), .56
bruceR::model_summary(), .56
bruceR::PROCESS(), .55, 56
car::recode(), .61
car::recode().61
compute(), 75
cor_diff, 4, 12
corr, 4, 11, 14, 54
count, 4
count (%%COMPUTE%%), 75
data.table::fread(), .37
data.table::fwrite(), .37
describe, 4, 12, 13, 54
dplyr::left_join(), .41
dtime, 15
EFA, 4, 6, 10, 15, 54
effectsize::sd_pooled(), 19
effectsize::t_to_d(), 19
EMMEANS, 4, 6, 18, 43, 44, 72
effsize::emmeans(), 18, 19
emmeans::emmeans(), 18, 19
emmeans::emmip(), 20, 44
emmeans::joint_tests(), 18, 19
effsize::summary(), 19
export, 4, 23, 38
foreign::read.dta(), .37
foreign::read.spss(), .37
format, 25, 26
formatF, 4, 25, 26
formatN, 4, 25, 25
formula_expand, 26
formula_paste, 27
Freq, 4, 27, 54
GGally::ggpairs(), .14
ggtext::element_markdown(), .69
ggtext::element_textbox(), .69
ggtext::geom_richtext(), .69
ggtext::geom_textbox(), .69
GLM_summary, 4, 28, 36, 48, 63
Glue, 4
Glue (Print), 52
glue::glue(), 53
glue::glue_col(), 53
glue::grand_mean_center(), 5, 29, 33
glue::grand_mean_center(), 5, 30, 32, 54
glue::grand_mean_center(), 5, 31, 54

81
group_mean_center, 5, 30, 32
haven::read_dta(), 37
haven::read_sav(), 37
haven::write_dta(), 23
haven::write_sav(), 23
HLM_ICC_rWG, 4, 33
HLM_summary, 4, 29, 35, 48, 63
import, 4, 24, 36
interactions::sim_slopes(), 56
lavaan, 39, 58
lavaan options, 10
lavaan::cfa(), 9, 10
lavaan::sem(), 56
lavaan_summary, 4, 10, 38, 48, 54, 59
lme4, 57
lmtest::grangertest(), 31
LOOKUP, 4, 40
MANOVA, 4, 6, 19–21, 42, 54, 72
MEAN, 4, 6, 18
MEAN (%COMPUTE%), 75
med_summary, 5, 46, 48, 54, 59
mediation, 46, 57, 58
mediation::mediate(), 46, 56
mixed.2_1b1w(bruceR-demodata), 6
mixed.3_1b2w(bruceR-demodata), 6
mixed.3_2b1w(bruceR-demodata), 6
MODE, 4
MODE (%COMPUTE%), 75
model_summary, 4, 28, 29, 35, 36, 47, 54, 59,
62, 63
MuMIn::r.squaredGLMM(), 47
MuMIn::std.coef(), 47
openxlsx::write.xlsx(), 23
p. 49
PCA, 4, 54
PCA (EFA), 15
performance::check_collinearity(), 48
performance::r2_mcfadden(), 47
performance::r2_nagelkerke(), 47
pkg_depend, 4, 51, 52
pkg_install_suggested, 4, 51, 52
Print, 4, 52
print_table, 4, 24, 29, 36, 48, 53, 63
PROCESS, 3, 39, 46, 48, 54, 55
psych::alpha(), 5, 6
psych::corr.test(), 12
psych::fa(), 15, 17
psych::kaiser(), 17
psych::omega(), 5, 6
psych::principal(), 15, 17
readxl::read_excel(), 37
RECODE, 4, 61
regress, 4, 29, 36, 62
rep_char, 63
RESCALE, 4, 64
RESCALE(). 66
RGB, 65
rio::export(), 23
rio::import(), 37
Run, 4, 65
scaler, 66
set.wd, 4, 67
set_wd(set.wd), 67
setwd, 67
show_colors, 5, 68
stats::p.adjust(), 11, 19
STD, 4
STD (%COMPUTE%), 75
SUM, 4
SUM (%COMPUTE%), 75
texreg::htmlreg(), 47, 48
texreg::screenreg(), 47, 48
theme_brace, 5, 69
tidyri::pivot_wider(), 44
TTEST, 4, 21, 44, 54, 71
utils::write.table(), 23
VAR, 30
vars::VAR(), 30
within.1(bruceR-demodata), 6
within.2(bruceR-demodata), 6
within.3(bruceR-demodata), 6
