Package ‘bruceR’

November 5, 2021

Title  Broadly Useful Convenient and Efficient R Functions
Version 0.7.3
Date 2021-11-05
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 where the current file is, print strings with rich formats and colors);
(2) multivariate computation
  (e.g., compute scale sums/means/... with reverse scoring);
(3) reliability and factor analyses;
(4) descriptive statistics and correlation analyses;
(5) multi-factor analysis of variance (ANOVA), simple-effect analysis, and post-hoc multiple comparison;
(6) tidy report of regression models and other results (to R Console and MS 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://github.com/psychbruce/bruceR
BugReports https://github.com/psychbruce/bruceR/issues
Depends R (>= 3.6.0)
Imports pacman, jmv, performance, glue, crayon, ggplot2, ggtext, cowplot, see
Suggests r Studioapi, rio, dplyr, tidyr, stringr, forcats, data.table, car, psych, afex, emmeans, effect size, mediation, interactions, lavaan, jtools, texreg, lme4, lmerTest, lmtest, vars, nnet, MuMIn, MASS, GGally
RoxygenNote 7.1.2
NeedsCompilation no
Repository CRAN
Date/Publication 2021-11-05 17:40:01 UTC

R topics documented:

bruceR-package .................................................. 3
Alpha ................................................................. 5
bruceR-demodata .................................................. 6
ccf_plot ............................................................. 6
CFA ................................................................. 8
Corr ................................................................. 9
cor_diff ............................................................ 11
Describe ............................................................ 12
dtime ............................................................... 13
EFA ................................................................. 14
EMMEANS ........................................................... 15
formatF ............................................................... 19
formatN ............................................................. 20
formula_expand .................................................... 20
formula_paste ...................................................... 21
Freq ................................................................. 22
GLM_summary ....................................................... 22
grand_mean_center ............................................... 23
granger_causality ............................................... 24
granger_test ......................................................... 25
group_mean_center ............................................... 26
HLM.ICC_rWG ...................................................... 27
HLM_summary ....................................................... 29
lavaan_summary .................................................... 32
LOOKUP ............................................................ 34
MANOVA ............................................................ 35
med_summary ....................................................... 38
model_summary ..................................................... 40
p ................................................................. 42
pkg_depend ......................................................... 43
pkg_install_suggested ......................................... 44
Print ............................................................... 44
print_table ......................................................... 45
PROCESS .......................................................... 47
RECODE ........................................................... 53
regress ............................................................ 53
rep_char ............................................................ 55
RESCALE .......................................................... 56
RGB .............................................................. 56
Run ................................................................. 57
**Description**

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

Install the latest development version by `devtools::install_github("psychbruce/bruceR")`

Check updates in **Release Notes**.

Report bugs in **GitHub Issues**.

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

**[Data]:**

- `rio`: Data import and export (for all file formats). *(import / export)*
- `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]:**

- `psych`: Toolbox for psychological and psychometric research.
- `emmeans`: Toolbox for estimated marginal means and contrasts.
- `effectsize`: Indices of effect size and standardized parameters.

**[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
   pkg_depend, pkg_install_suggested
   formatF, formatN
   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
   CFA

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

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

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

(7) Mediation and Moderation Analyses  PROCESS
   lavaan_summary
   med_summary

(8) Additional Toolbox for Statistics and Graphics  grand_mean_center
   group_mean_center
   ccf_plot
   granger_test
   granger_causality
   theme_bruce
   show_colors

Note

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

Author(s)

Han-Wu-Shuang (Bruce) Bao
E-mail: <baohws@foxmail.com>
Description
An extension of \texttt{jmv::reliability()}. It reports (1) scale reliability statistics (Cronbach's $\alpha$ and McDonald's $\omega$) and (2) item reliability statistics (item-rest correlation [i.e., corrected item-total correlation] and what Cronbach's $\alpha$ and McDonald's $\omega$ would be if the item was dropped).

Three options to specify the variable list:
1. \texttt{var + items}: use the common and unique parts of variable names.
2. \texttt{vars}: directly define a variable list.
3. \texttt{varrange}: use the start and end positions of a variable list.

Usage
\texttt{Alpha(data, var, items, vars = NULL, varrange = NULL, rev = NULL)}

Arguments
data \hspace{1cm} Data frame.
var \hspace{1cm} \textbf{[option 1]} Common part across multiple variables (e.g., "RSES", "SWLS").
items \hspace{1cm} \textbf{[option 1]} Unique part across multiple variables (e.g., 1:10).
vars \hspace{1cm} \textbf{[option 2]} Character vector specifying a variable list (e.g., \texttt{c("E1", "E2", "E3", "E4", "E5")}).
varrange \hspace{1cm} \textbf{[option 3]} Character with ":" specifying the start and end positions of a variable list (e.g., ":A1:E5").
rev \hspace{1cm} \textbf{[optional]} Reverse-scoring variables. It can be (1) a numeric vector specifying the positions of reverse-scoring variables (not recommended) or (2) a character vector directly specifying the variable list (recommended).

Value
A result object obtained from \texttt{jmv::reliability()}. 

See Also
\texttt{MEAN}

Examples
# ?psych::bfi
\texttt{Alpha(bfi, "E", 1:5)} # "E1" & "E2" should be reverse scored
\texttt{Alpha(bfi, "E", 1:5, rev=1:2)} # correct
\texttt{Alpha(bfi, "E", 1:5, rev=c("E1", "E2"))} # also correct
\texttt{Alpha(bfi, vars=c("E1", "E2", "E3", "E4", "E5"), rev=c("E1", "E2"))}
\texttt{Alpha(bfi, varrange="E1:E5", rev=c("E1", "E2"))}
# using dplyr::select()
bfi %>% select(E1, E2, E3, E4, E5) %>%
  Alpha(vars=names(.), rev=c("E1", "E2"))

---

**Demo data.**

**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

---

**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

```r
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.
- **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
granger_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 jmv::cfa() and lavaan::cfa().

Usage

CFA(
data,
  model = "A =~ a[1:5]; B =~ b[c(1,3,5)]; C =~ c1 + c2 + c3",
  highorder = ",",
  orthogonal = FALSE,
  missing = "listwise",
  style = "lavaan",
  CI = FALSE,
  MI = FALSE
)

Arguments
data Data frame.
model Model formula. See examples.
highorder High-order factor. Default is ",".
orthogonal Default is FALSE. If TRUE, all covariances among latent variables are set to zero, and only "lavaan" style will be output.
missing
style
CI
MI
Value
See Also
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="
  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]")
Corr

```r
data = NULL,
method = "pearson", # default, "spearman", or "kendall"
p.adjust = "none", # default, "fdr", "holm", "bonferroni"
all.as.numeric = TRUE, # default or FALSE. Transform all variables into numeric (continuous).
digits = 2
file = NULL, # default, plot in RStudio or a file name (.doc).
plot = TRUE, # default or FALSE. Plot the correlation matrix.
plot.range = c(-1, 1),
plot.palette = NULL, # default c("#B52127","white","#2171B5")
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`: 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.
- `plot.range`: Range of correlation coefficients for plot. Default is c(-1,1).
- `plot.palette`: Color gradient for plot. Default is c("#B52127","white","#2171B5"). You may also set it to, e.g., c("red","white","blue").
- `plot.color.levels`: Default is 201.
- `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 the correlation results obtained from `psych::corr.test()`.

**See Also**
- `Describe`

**Examples**
```r
Corr(airquality)
Corr(airquality, p.adjust="bonferroni")
```
```r
d=as.data.table(psych::bfi)
d[,"m:="("
cor_diff

gender=as.factor(gender),
education=as.factor(education),
E=MEAN(d, "E", 1:5, rev=c(1,2), likert=1:6),
A=MEAN(d, "A", 1:5, rev=1, likert=1:6),
C=MEAN(d, "C", 1:5, rev=c(4,5), likert=1:6),
N=MEAN(d, "N", 1:5, likert=1:6),
O=MEAN(d, "O", 1:5, rev=c(2,5), likert=1:6)
]
Corr(d[,.(age, gender, education, E, A, C, N, O)])

---

cor_diff  Test the difference between two correlations.

Description
Test the difference between two correlations.

Usage

```r
cor_diff(r1, n1, r2, n2, n = NULL, rcov = NULL)
```

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

```r
# 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 (to R Console or MS Word).

Description

Descriptive statistics (to R Console or MS Word).

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
)

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 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 "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 ggplot2 object if users set plot=TRUE.
### dtime

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**

```r
t0 = Sys.time()
dtime(t0)
```

---

**EFA**  
*Exploratory factor analysis (EFA).*

**Description**

An extension of `jmv::efa()`.

**Usage**

```r
EFA(
  data,
  vartext,
  method = "eigen",
  extraction = "pa",
  rotation = "varimax",
  nFactors = 1,
  hideLoadings = 0.3
)
```

**Arguments**

- `data`  
  Data frame.

- `vartext`  
  Character string specifying the model (e.g., "X[1:5] + Y[c(1,3)] + Z").

- `method`  
  "eigen" (default), "parallel", or "fixed", the way to determine the number of factors.

- `extraction`  
  "pa" (default), "ml", or "minres", using "principal axis", "maximum likelihood", or "minimum residual" as the factor extraction method, respectively.

- `rotation`  
  "varimax" (default), "oblimin", or "none", the rotation method.

- `nFactors`  
  An integer (default is 1) fixing the number of factors. Only relevant when `method="fixed"`.

- `hideLoadings`  
  A number (0~1, default is 0.3) for hiding factor loadings below this value.

**Value**

No return value.

**Note**

It does not have the extraction method "Principal Components". You may still use SPSS.
See Also

jmv::efa()

Examples


Description

Easily 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 \( \geq 3 \) levels.

This function is based on and extends the (1) emmeans::joint_tests(), (2) emmeans::emmeans(), and (3) emmeans::contrast() functions. You only need to specify the model object, to-be-tested effect(s), and moderator(s). Almost all results you need will be displayed in an elegant manner, 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.

To compute Cohen’s \( d \) and its 95% CI in pairwise comparisons, this function uses the pooled SD: \( SD_{pooled} = \sqrt{MSE} \), where MSE is of the effect term extracted from ANOVA table.

Disclaimer: There is substantial disagreement on what is the appropriate pooled SD to use in computing effect sizes. 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

EMMEANS(
  model,
  effect = NULL,
  by = NULL,
  contrast = "pairwise",
  reverse = TRUE,
  p.adjust = "bonferroni",
  sd.pooled = NULL,
  spss = TRUE,
  digits = 2,
  nsmall = digits
)
Arguments

- **model**: A model object returned by `MANOVA`.
- **effect**: The 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", "mtv", "dunnettx", "sidak", "scheffe", "bonferroni". For details, see `stats::p.adjust()` and `emmeans::summary()`.
- **sd.pooled**: By default, it uses \( \sqrt{\text{MSE}} \) to compute Cohen’s \( d \). Users may also manually set it (e.g., the \( SD \) of a reference group, or using `effectsize::sd_pooled()`).
- **spss**: Return results identical to SPSS. Default is TRUE, which uses the `lm` (rather than `aov`) object in `model` for `emmeans::joint_tests()` and `emmeans::emmeans()`.
- **digits, nsmall**: Number of decimal places of output. Default is 2.

Value

The same object as returned by `MANOVA` (for recursive use).

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 ANOVA, we call it "simple-effect analysis"; in regression, we also call it "simple-slope analysis". They are identical in statistical principles. Nonetheless, the situations in ANOVA can be a bit more complex because we sometimes have a three-factors design.

In a regular two-factors design, we only test "**simple main effects**". That is, on the 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 effects**" and "**simple simple effects**". That is, on the different combinations of levels of factors "B" and "C", the main effects of "A" would be different.
In SPSS, we usually use the MANOVA and/or the GLM + /EMMEANS syntax to perform such analyses. Tutorials (in Chinese) for the SPSS syntax can be found in: Tutorial #1, Tutorial #2, and Tutorial #3.

Here, the R function EMMEANS can do the same thing as in SPSS and can do much better and easier (just see the section "Examples").

To note, simple effects per se do NOT need any form of 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 and softwares, "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. For example, we can perform the post-hoc comparisons of mean values (1) across multiple levels of one factor in a pairwise way or (2) particularly between the two conditions "A1B1" and "A2B2".

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 post-hoc tests by its nature but do NOT have any relationship with simple-effect analyses. In other words, "(post-hoc) multiple comparisons" are independent of "interaction effects" and "simple effects". What’s more, when the simple main effect is of a factor with 3 or more levels, we also need to do multiple comparisons (e.g., pairwise comparisons) within the simple-effect analysis. In this situation (i.e., >= 3 levels), we need p-value adjustment methods such as Bonferroni, Tukey’s HSD (honest significant difference), FDR (false discovery rate), and so forth.

There are many ways to do multiple comparisons. All these methods are included in the current EMMEANS function. If you are familiar with SPSS syntax, you may feel that the current R functions MANOVA and EMMEANS are a nice combination of the SPSS syntax MANOVA and GLM + /EMMEANS. Yes, they are. More importantly, they outperform the SPSS syntax, either for its higher convenience or for its more fruitful results.

- "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

MANOVA, bruceR-demodata

Examples

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

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

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

between.3
MANOVA(data=between.3, dv="SCORE", between=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 can test many other combinations of effects

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

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

within.2
MANOVA(data=within.2, dvs="A1B1:A2B3", dvs.pattern="A(.)B(.)", 
       within=c("A", "B") %>%
  EMMEANS("A", by= "B") %>%
  EMMEANS("B", by= "A") # singular error matrix

within.3
       within=c("A", "B", "C") %>%
  EMMEANS("A", by= "B") %>%
  EMMEANS(c("A", "B"), by= "C") %>%
  EMMEANS("A", by= c("B", "C"))

#### Mixed Design ####

mixed.2_1b1w
MANOVA(data=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(data=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"))

mixed.3_2b1w
MANOVA(data=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"))

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

---

**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

```r
formatF(pi, 20)
```

---

### formatN

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

**Usage**

```r
formatN(x, mark = ",")
```

**Arguments**

- `x`: A number or numeric vector.
- `mark`: Usually ",".

**Value**

Formatted character string.

**See Also**

`format`, `formatF`

**Examples**

```r
formatN(1234)
```

---

### formula_expand

**Expand all interaction terms in a formula.**

**Description**

Expand all interaction terms in a formula.

**Usage**

```r
formula_expand(formula, as.char = FALSE)
```
Arguments

- **formula**: R formula or a character string indicating the formula.
- **as.char**: Return character? Default is FALSE.

Value

A formula/character object including all expanded terms.

Examples

```
formula_expand(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 (to R Console or MS Word).*

**Description**  
Frequency statistics (to R Console or MS Word).

**Usage**  
Freq(var, label = NULL, sort = "", digits = 1, nsmall = digits, file = NULL)

**Arguments**
- **var**  
  Vector or variable.
- **label**  
  [optional] A vector re-defining the labels of values.
- **sort**  
  "" (default, sorted by raw order), "-" (decreasing), or "+" (increasing).
- **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**
- Freq(bfi$education)
- Freq(bfi$gender, label=c("Male", "Female"))
- Freq(bfi$age)

---

**GLM_summary**  
*Tidy report of GLM (lm and glm models).*

**Description**  
Tidy report of GLM (lm and glm models).

**Usage**  
GLM_summary(
  model,
  robust = FALSE,
  cluster = NULL,
  digits = 3,
  nsmall = digits,
  ...
)

Arguments

model  A model fitted by 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.

*** "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. If you specify cluster, you may also specify the type of robust. If you do not specify robust, "HC1" will be set as default.

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

...  Other parameters. You may re-define formula, data, or family.

Value

No return value.

See Also

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")

---

description

**Description**

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

```r
grand_mean_center(data, vars = names(data), std = FALSE, add_suffix = "")
```

**Arguments**

- `data` Data object.
- `vars` Variable(s) to be centered.
- `std` Standardized or not. Default is `FALSE`.
- `add_suffix` The suffix of the centered variable(s). Default is "". You may set it to ",", ",center", etc.

**Value**

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

**See Also**

`group_mean_center`

**Examples**

```r
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**

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

- **varmodel**: VAR model fitted using the `vars::VAR()` function.
- **var.y, var.x**: [optional] Default is NULL (all variables). If specified, then perform tests for specific variables. Values can be a single variable (e.g., "X"), a vector of variables (e.g., c("X1","X2")), or a string containing regular expression (e.g., "X1|X2").
- **test**: F test and/or Wald $\chi^2$ test. Default is both: c("F","Chisq").
- **file**: File name of MS Word (.doc).
- **check.dropped**: Check dropped variables. Default is FALSE.

Details

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:
# "vars" package should be installed and loaded.
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

- `formula`: Model formula like `y ~ x`.
- `data`: Data frame.
- `lags`: Time lags. Default is `1:5`.
- `test.reverse`: Whether to test reverse causality. Default is `TRUE`.
- `file`: File name of MS Word (.doc).

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

```r
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 test
```

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"
)
```
HLM_ICC_rWG

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).

Value

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

See Also

- `grand_mean_center`

Examples

```r
# Load data
data <- data.table(x=1:9, g=rep(1:3, each=3))

# Center variable x by group
centered_data <- group_mean_center(data, "x", by="g")
centered_data

# Center variable x by group with suffix ".c"
centered_data_c <- group_mean_center(data, "x", by="g", add_suffix=".c")
centered_data_c
```

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

```r
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).
- **rwg.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, $\sigma_E^2$), it is required to specify which type of uniform distribution is.
  - For *continuous* uniform distribution, $\sigma_E^2 = (\text{max} - \text{min})^2 / 12$. Then rwg.levels is not useful and will be set to 0 (the default).
  - For *discrete* uniform distribution, $\sigma_E^2 = (\text{A}^2 - 1) / 12$, 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

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

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

$$\text{ICC}(1) = \frac{\text{var.u0}}{(\text{var.u0} + \text{var.e})} = \frac{\sigma_{u0}^2}{(\sigma_{u0}^2 + \sigma_e^2)}$$

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)

$$\text{ICC}(2) = \frac{\text{mean(var.u0} / (\text{var.u0} + \text{var.e} / n.k)) = \Sigma(\sigma_{u0}^2 / (\sigma_{u0}^2 + \sigma_e^2 / n_k))/K$$

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)

$$\text{rWG} = 1 - \frac{\sigma_e^2}{\sigma_{EU}^2}$$

$$\text{rWG(J)} = 1 - \frac{\sigma_{MJ}^2 / \sigma_{EU}^2}{[J * (1 - \sigma_{MJ}^2 / \sigma_{EU}^2) + \sigma_{MJ}^2 / \sigma_{EU}^2]}$$

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

Invisibly return a list of results.

### References


### See Also

R package "multilevel"

### Examples

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

# 7-point scale
data = lmerTest::carrots
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)
```

---

### Description

Nice report of **Hierarchical Linear Model (HLM)**, also known as **Multilevel Linear Model (MLM)** or **Linear Mixed Model (LMM)**. HLM, MLM, or LMM (the same) refers to a model with nested data (e.g., Level-1: participants, Level-2: city; or Level-1: repeated-measures within a participant, Level-2: participants).

### Usage

```r
HLM_summary(
    model = NULL,
    level2.predictors = NULL,
    vartypes = NULL,
    test.rand = FALSE,
    digits = 3,
    nsmall = digits,
)```
Arguments

model A model fitted by lmer or glmer function using the lmerTest package.

level2.predictors

[only for lmer] [optional] Default is NULL. If you have predictors at level 2, besides putting them into the formula in the lmer function as usual, you may also define here the level-2 grouping/clustering variables and corresponding level-2 predictor variables.

*** Example: level2.predictors="School: W1 + W2; House: 1", where School and House are two grouping variables, W1 & W2 are school-level predictors, and there is no house-level predictor.

*** If there is no level-2 predictor in the formula of lmer, just leave this parameter blank.

vartypes [only for lmer] Manually setting variable types. Needless in most situations.

test.rand [only for lmer] 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. But for some statistics (e.g., R^2, ICC), to provide more precise information, we fix the decimal places to 5.

... Other optional parameters. You may re-define formula, data, or family.

Details

Hierarchical Linear Model (HLM), aka. Multilevel Linear Model (MLM) or Linear Mixed Model (LMM), is more complex than General Linear Model (GLM; i.e., OLS regression). Predictor variables at different levels may have five types:

1. Intercept The overall intercept (γ00)
2. L1fixed Level-1 predictor with fixed slope
3. L1random-GROUP-L1VAR Level-1 predictor with random slopes nested with a grouping/clustering variable
4. L2-GROUP Level-2 predictor (e.g., GDP per capita at city level), always with fixed slope unless there is also a level-3 structure.

*** NOTE: the current version of ‘HLM_summary’ function does not consider three-levels design, so you may only use this function in two-levels HLM or cross-classified HLM.

5. Cross-GROUP-L1VAR Cross-level interaction consisting of level-1 and level-2 predictors

The degrees of freedom (df) of predictor variables in HLM vary across different levels and also depend on the variable types. However, different software use different estimation methods and thus provide somewhat different dfs, which may be confusing. Whereas the lmerTest package in R provides dfs that are estimated by the Satterthwaite’s (1946) approximation (i.e., a data-driven approach without defining variable types), the HLM software provides dfs that totally depend on the variable types (i.e., a theory-driven approach).
Value

No return value.

References


See Also

GLM_summary, regress

Examples

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, level2.predictors="Consumer: Age + Frequency")
HLM_summary(hlm.2, level2.predictors="Consumer: Age + Frequency")
anova(hlm.1, hlm.2)
lavaan_summary

Tidy report of lavaan model.

Description

Tidy report of lavaan model.

Usage

lavaan_summary(
  lavaan,  # Model object fitted by lavaan.
  ci = c("raw", "boot", "bc.boot", "bca.boot"),
  nsim = 100,  # Number of simulation samples (bootstrap resampling) for estimating SE and 95% CI of user-defined parameter(s). Default is 100 for running examples faster. In formal analyses, however, nsim=1000 (or larger) is strongly suggested!
  seed = NULL,  # Random seed for obtaining reproducible results. Default is NULL.
  digits = 3,  # Number of decimal places of output. Default is 3.
  nsmall = digits,  # File name of MS Word (.doc).
  print = TRUE,  # Print results. Default is TRUE.
  file = NULL)

Arguments

lavaan  # Method for estimating the standard error (SE) and 95% confidence interval (CI) of user-defined parameter(s). Default is "raw" (the standard approach of lavaan). Other options include:
    "boot" Percentile Bootstrap
    "bc.boot" Bias-Corrected Percentile Bootstrap
    "bca.boot" Bias-Corrected and Accelerated (BCa) Percentile Bootstrap

Value

Invisibly return a list of results:

fit  # Fit measures.
path  # Path coefficients.
effect  # Used-defined effect estimates.
See Also
PROCESS

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

lv=lavaan::sem(model=model, data=airquality)
lavaan::summary(lv, 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.

**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 parameter 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

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

MANOVA

Multi-factor ANOVA.

Description

Easily perform multi-factor ANOVA (between-subjects, within-subjects, and mixed designs), with or without covariates (ANCOVA). Print results to R Console (and MS Word).

This function is based on and extends the afex::aov_ez() function. 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 in an elegant manner, including effect sizes (partial \(\eta^2\)) and their confidence intervals (CIs). 90% CIs for partial \(\eta^2\) are reported, following the suggestion by Steiger (2004).

Usage

MANOVA(
  data,
  subID = NULL,
  dv = NULL,
  dvs = NULL,
  dvs.pattern = "",
  between = NULL,
within = NULL,
covariate = NULL,
sph.correction = "none",
digits = 2,
nsmall = digits,
file = NULL
)

Arguments

data Data frame. Both long-format and wide-format can be used.

• If using long-format data, please also set subID.
• If using wide-format data (i.e., one subject occupies one row, and repeated
measures occupy multiple columns), the function can automatically trans-
form the data into long-format.

subID Subject ID.

• If using long-format data, you should set the subject ID.
• If using wide-format data, no need to set this parameter.

dv Variable name of dependent variable.

• If using long-format data, then dv is the outcome variable.
• If using wide-format data, then dv can only be used for complete between-
subjects design. For designs with repeated measures, please use dvs and
dvs.pattern.

dvs [only for "wide-format" data and designs with repeated measures]

Variable names of repeated measures.

• You can use ":" to specify a range of variables: e.g., "A1B1:A2B3" (similar
to the SPSS syntax "TO"; the variables should be put in order)
• You can also use a character vector to specify variable names: e.g., c("Cond1","Cond2","Cond3")

dvs.pattern [only for "wide-format" data and designs with repeated measures]
If you set dvs, you must also set the pattern of variable names by using regular
expressions.

Examples:

• "Cond(.)" can extract levels from "Cond1","Cond2","Cond3",...

You can rename the factor name by using within: e.g., within="Condition"

• "X(.+)Y(.+)" can extract levels from "X01Y01","X02Y02","XaaYbc",...
• "X(.+)Y(.+)" can extract levels from "X1Y1","XaYb","XaY002",...

Tips on regular expression:

• "(.*" extracts any single character (can be number, letter, or other sym-

bols)
• "(.+)^ extracts >= 1 character(s)
• "(.+)^ extracts >= 0 character(s)
• "([0-9])" extracts any single number
• "([a-z])" extracts any single letter
MANOVA

- each pair of "()" extracts levels for each factor

between

Between-subjects factors. Character string (e.g., "A") or vector (e.g., c("A", "B")). Default is NULL.

within

Within-subjects factors. Character string (e.g., "A") or vector (e.g., c("A", "B")). Default is NULL.

covariate

Covariates (if necessary). Character string (e.g., "age") or vector (e.g., c("gender", "age", "edu")). Default is NULL.

sph.correction

[only effective for repeated measures with >= 3 levels]

Sphericity correction method to adjust 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).

digits, nsmall

Number of decimal places of output. Default is 2.

file

File name of MS Word (.doc).

Value

A result object returned by afex::aov_ez().

References


See Also

EMMEANS, bruceR-demodata

Examples

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

between.1

MANOVA(data=between.1, dv="SCORE", between="A")

between.2

MANOVA(data=between.2, dv="SCORE", between=c("A", "B"))

between.3

MANOVA(data=between.3, dv="SCORE", between=c("A", "B", "C"))

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

within.1

MANOVA(within.1, dvs="A1:A4", dvs.pattern="A(.)")
within="A")

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

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

within.3

#### Mixed Design ####

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

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

mixed.3_2b1w
MANOVA(data=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=data.new, dvs="Cond_01:Cond_04", dvs.pattern="Cond_(.)", between="Group", within="Condition") # renamed the within-subjects factor

?afex::obk.long
MANOVA(data=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 (to R Console or MS Word).

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

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)
### Monte Carlo simulation (quasi-Bayesian approximation)
### (bootstrap method is not applicable to "lmer" models)

### Hypothesis: Crips → Sweetness → Preference (for carrots)
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)
model_summary

Tidy report of regression models (to R Console or MS Word).

Description

Tidy report of regression models (to R Console or MS Word). Most types of regression models are supported! This function is an extension (and combination) of `texreg::screenreg()`, `texreg::htmlreg()`, `MuMIn::std.coef()`, `MuMIn::r.squaredGLMM()`, `performance::r2_mcfadden()`, `performance::r2_nagelkerke()`.

Usage

```r
model_summary(
  model_list,
  std = FALSE,  # Standardized coefficients? Default is FALSE. Only applicable to linear models and linear mixed models. Not applicable to generalized linear (mixed) models.
  digits = 3,  # Number of decimal places of output. Default is 3.
  nsmall = digits,  # Equivalent to nsmall.
  file = NULL,  # File name of MS Word (.doc).
  zero = ifelse(std, FALSE, TRUE),  # Display "0" before "."? Default is TRUE.
  modify_se = NULL,  # 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 = NULL,  # Replace model names.
  line = TRUE,  # Lines look like true line (TRUE) or === ---=== (FALSE). Only relevant to R Console output.
  bold = 0,  # The p-value threshold below which the coefficients will be formatted in bold.
  ...  # Other parameters passed to `texreg::screenreg()` or `texreg::htmlreg()`.
)
```

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).
- `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 parameters passed to `texreg::screenreg()` or `texreg::htmlreg()`.
**model_summary**

**Value**

Invisibly return the output (character string).

**See Also**

- PROCESS
- GLM_summary
- HLM_summary
- med_summary
- lavaan_summary
- print_table

**Examples**

#### Example 1: Linear Model ####

```r
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")
unlink("OLS Models.doc")  # delete file for test
```

#### 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 test
```

#### 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 test
```

#### 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")
```
#### 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 test
```

---

**p**

*Compute p value.*

**Description**

Compute p value.

**Usage**

```r
p(  
z = NULL,
t = NULL,
f = NULL,
r = NULL,
chi2 = NULL,
n = NULL,
df = NULL,
df1 = NULL,
df2 = NULL,
digits = 2,
nsmall = 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, chi2: z, t, F, r, \( \chi^2 \) value.
- n, df, df1, df2: Sample size or degree of freedom.
- 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

- \( p.z(1.96) \)
- \( p.t(2, 100) \)
- \( p.f(4, 1, 100) \)
- \( p.r(0.2, 100) \)
- \( p.chi2(3.84, 1) \)

```
p(z=1.96)  # Two-tailed p value of z
p(t=2, df=100)  # Two-tailed p value of t
p(f=4, df1=1, df2=100)  # One-tailed p value of F
p(r=0.2, n=100)  # Two-tailed p value of r
p(chi2=3.84, df=1)  # One-tailed p value of \( \chi^2 \)
```
pkg_install_suggested

Install suggested R packages.

Description
It checks and installs R packages suggested by bruceR (default) or any other package.

Usage
pkg_install_suggested(by = "bruceR")

Arguments
by
Suggested by which package? Default is "bruceR".

Value
No return value.

See Also
pkg_depend

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(...)

Value
A character vector of package names.

See Also
pkg_install_suggested
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)

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 in bruceR. The implementation of Word output is using HTML code. You can check the raw HTML code by opening the Word file with any text editor. See here for a list of other functions in bruceR that support Word output.
print_table

Usage

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

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

Describe, Freq, Corr
**Examples**

```r
print_table(airquality, file="airquality.doc")
unlink("airquality.doc")  # delete file for test

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 test
```

---

**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 first builds regression models according to the data, variable names, and a few other parameters 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 automatically conduct mean-centering before model building.

Then, it 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: (1) regression model summary (using bruceR::model_summary() to summarize the models) and (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,
  ci = c("boot", "bc.boot", "bca.boot", "mcmc"),
  nsim = 100,
  seed = NULL,
  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 parameter 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) level-2 cluster(s): e.g., "School_ID" or 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_ID)" or "(1 | Sub) + (1 | Item)". You can set these parameters to include more complex terms (e.g., random slopes). In most cases, no need to set these parameters.

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 parameter.

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 parameter.

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=5201314, 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 have 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).

std 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

### PROCESS Analyses ###

#### Demo Data ####

```r
data = mediation::student %>%
dplyr::select(SCH_ID, free, morale, 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 ####

```r
Freq(data$gender)
Freq(data$pass)
Describe(data)  # file="xxx.doc"
Corr(data[,4:11])  # file="xxx.doc"
```

#### PROCESS Analyses ####

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

# (Johnson-Neyman (J-N) interval and plot)
PROCESS(data, y="score", x="gender", mods="late") -> P
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),  # moderator 1: late
       mod2.val = seq(1, 15, 2),  # moderator 2: family_inc
       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",
       ci="mcmc", nsim=100, seed=1)

## Model 6 ##
PROCESS(data, y="score", x="parent_edu",
       meds=c("family_inc", "late"),
       covs=c("gender", "partjob"),
       med.type="serial",
       ci="boot", 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
RECODE

Recode a variable.

Description
Based on `car::recode()`.

Usage
RECODE(var, recodes)

Arguments
- `var` Variable (numeric, character, or factor).
- `recodes` Character string: 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

d[, "var.recoded" := RECODE(var, "lo:1=0; c(2,3)=1; 4=2; 5:hi=3; else=999")]
d
```

regress

Regression analysis.

Description
Regression analysis.

Usage
regress(
  formula,
  data,
  family = NULL,
  digits = 3,
  nsmall = digits,
  robust = FALSE,
  cluster = NULL,
)
Arguments

formula  Model formula like \( y \sim x_1 + x_2 \) (for \texttt{lm}, \texttt{glm}) or \( y \sim x_1 + x_2 + (1 | \text{group}) \) (for \texttt{lmer}, \texttt{glmer}).
data  Data frame.
family  [optional] The same as in \texttt{glm} and \texttt{glmer} (e.g., family=\texttt{binomial} will fit a logistic model).
digits  Number of decimal places of output. Default is 3.
nsmall  Number of decimal places of output. Default is 3.
robust  [only for \texttt{lm} and \texttt{glm}] False (default), \texttt{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 \texttt{?sandwich::vcovHC} and \texttt{?jtools::summ.lm}. *** "HC1" is the default of Stata, whereas "HC3" is the default suggested by the \texttt{sandwich} package.
cluster  [only for \texttt{lm} and \texttt{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. If you specify \texttt{cluster}, you may also specify the type of \texttt{robust}. If you do not specify \texttt{robust}, "HC1" will be set as default.
level2.predictors  [only for \texttt{lmer}] [optional] Default is \texttt{NULL}. If you have predictors at level 2, besides putting them into the formula in the \texttt{lmer} function as usual, you may also define here the level-2 grouping/clustering variables and corresponding level-2 predictor variables.
  *** Example: level2.predictors="School: W1 + W2; House: 1", where School and House are two grouping variables, W1 & W2 are school-level predictors, and there is no house-level predictor.
  *** If there is no level-2 predictor in the formula of \texttt{lmer}, just leave this parameter blank.
vartypes  [only for \texttt{lmer}] Manually setting variable types. Needless in most situations.
test.rand  [only for \texttt{lmer}] \texttt{TRUE} or \texttt{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.
**Examples**

```r
## lm
regress(Temp ~ Month + Day + Wind + Solar.R, data=airquality, robust=TRUE)

## glm
regress(case ~ age + parity + education + spontaneous + induced,
        data=infert, family=binomial, robust="HC1", cluster="stratum")

## lmer
library(lmerTest)
regress(Reaction ~ Days + (Days | Subject), data=sleepstudy)
regress(Preference ~ Sweetness + Gender + Age + Frequency +
        (1 | Consumer), data=carrots)

## glmer
library(lmerTest)
data.glmm=MASS::bacteria
regress(y ~ trt + week + (1 | ID), data=data.glmm, family=binomial)
regress(y ~ trt + week + hilo + (1 | ID), data=data.glmm, family=binomial)
```

---

**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**

```r
rep_char(char, rep.times)
```

**Arguments**

- `char` Character string.
- `rep.times` Times for repeat.

**Value**

Character string.

**Examples**

```r
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

```r
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

```r
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.
### Run

#### Value

"#rrggbb" or "#rrggbbaa".

#### Examples

```
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

```
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

```r
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

```r
scaler(1:5)
# the same: RESCALE(1:5, to=0:1)
```

set.wd  

Set working directory to where the current file is.

Description

Set working directory to the path of the currently opened file. 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, directly = TRUE, 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.
- **directly**: TRUE (default) or FALSE. Default is to directly execute `setwd("...")` within the function (recommended). Otherwise, it will send code `setwd("...")` to the R Console and then execute it (not recommended due to a delay of execution).
- **ask**: TRUE or FALSE (default). If TRUE, you can select a folder with the prompt of a dialog.

Value

Invisibly return the path.

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 directory
set.wd("../") # set working directory to the parent directory
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
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 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**<sup>2</sup>") in plot text or title, and this function draws text elements with rich text format.

For more usage, see:

• ggtext::geom_richtext()
• ggtext::geom_textbox()
• ggtext::element_markdown()
• ggtext::element_textbox()

Usage
theme_bruce(
  markdown = FALSE,
  base.size = 12,
  line.size = 0.5,
  border = "black",
  bg = "white",
  panel.bg = "white",
  tag = "bold",
  plot.title = "bold",
)
theme_bruce

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

markdown Use element_markdown() instead of element_text(). Default is FALSE. If set to TRUE, then you should also use element_markdown() in theme() (if any).

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=data.table(bfi)
d[, := (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\text{<sub>Big 5</sub>}",
       y="Openness\text{<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) +
  geom_errorbar(aes(x=X1, ymin=Y.mean-Y.se, ymax=Y.mean+Y.se),
                width=0.1, color="black", position=position_dodge(0.6)) +
  scale_y_continuous(expand=expansion(add=0),
                    limits=c(0,8), breaks=0:8) +
  scale_fill_brewer(palette="Set1") +
  labs(x="Independent Variable (*X*)", # italic X
       y="Dependent Variable (*Y*)", # italic Y
       title="Demo Plot<sup>bruceR</sup>'") +
  theme_bruce(markdown=TRUE, border="")
```

%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

```r
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

```r
3:4 %anyin% 1:3 # TRUE
4:5 %anyin% 1:3 # FALSE
```
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 the variable list:

1. `var + items`: use the common and unique parts of variable names.
2. `vars`: directly define a variable list.
3. `varrange`: use the start and end positions of a variable list.

Usage

```r
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
)
```
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] Common part across multiple variables (e.g., "RSES", "SWLS").

items
  [option 1] Unique part across multiple variables (e.g., 1:10).

vars
  [option 2] Character vector specifying a variable list (e.g., c("E1","E2","E3","E4","E5")).

varrange
  [option 3] Character with ":" specifying the start and end positions of a variable list (e.g., "A1:E5").

data
  [only for COUNT] The value to be counted.

rev
  [optional] Reverse-scoring variables. It can be (1) a numeric vector specifying the positions of reverse-scoring variables (not recommended) or (2) a character vector directly specifying the variable list (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

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")
]

data=as.data.table(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

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.
\%notin\%

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

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

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

pattern  Character string containing regular expressions to be matched.
vector  Character vector.

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

%^%  

Paste strings together.

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%%, 64
%^%, 4, 68
%allin%, 4, 62, 63, 67, 68
%anyin%, 4, 63, 66, 68
%in%, 63, 67, 68
%nonein%, 4, 63, 66, 68
%notin%, 4, 67
%partin%, 4, 63, 67, 68
afex::aov_ez(), 35, 37
Alpha, 4, 5, 64
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::PROCESS(), 47
car::recode(), 53
ccf_plot, 4, 6, 25, 26
CFA, 4, 8
CONSEC, 4
CONSEC (%COMPUTE%), 64
corr_diff, 4, 11
Corr, 4, 9, 13, 46
COUNT, 4
COUNT (%COMPUTE%), 64
Describe, 4, 10, 12, 46
dplyr::left_join(), 35
dtime, 13
EFA, 4, 14
effectsize::sd_pooled(), 16
effectsize::t_to_d(), 15
EMMEANS, 4, 6, 15, 37
emeans::contrast(), 15
emeans::eff_size(), 15
emeans::emmeans(), 15, 16
emeans::joint_tests(), 15, 16
emeans::summary(), 16
export, 3
format, 19, 20
formatF, 4, 19, 20
formatN, 4, 19, 20
formula_expand, 20
formula_paste, 21
Freq, 4, 22, 46
GGally::ggpairs(), 12
ggtext::element_markdown(), 60
ggtext::element_textbox(), 60
ggtext::geom_richtext(), 60
ggtext::geom_textbox(), 60
GLM_summary, 4, 22, 31, 41
Glue, 4
Glue (Print), 44
glue::glue(), 45
.glue:glue_col(), 45
grand_mean_center, 4, 23, 27
granger_causality, 4, 24, 26
granger_test, 4, 8, 25, 25
group_mean_center, 4, 24, 26
HLM_ICC_rWG, 4, 27
HLM_summary, 4, 23, 29, 41
import, 3
interactions::sim_slopes(), 47
jmv::cfa(), 8, 9
jmv::efa(), 14, 15
jmv::reliability(), 5
lavaan, 32, 50
lavaan::cfa(), 8, 9
lavaan::sem(), 47
lavaan_summary, 4, 32, 41, 51
INDEX

lme4, 49
lmtest::grangertest(), 25
LOOKUP, 4, 34

MANOVA, 4, 6, 16, 17, 35
MEAN, 4, 5
MEAN(%%COMPUTE%%), 64
med_summary, 4, 38, 41, 51
mediation, 38, 49, 50
mediation::mediate(), 39, 47
mixed.2_1bw(bruceR-demodata), 6
mixed.3_1b2w(bruceR-demodata), 6
mixed.3_2b1w(bruceR-demodata), 6
MODE, 4
MODE(%%COMPUTE%%), 64
model_summary, 4, 40, 51
MuMIn::r.squaredGLMM(), 40
MuMIn::std.coef(), 40

p, 42
performance::r2_mcfadden(), 40
performance::r2_nagelkerke(), 40
pkg_depend, 4, 43, 44
pkg_install_suggested, 4, 44, 44
Print, 4, 44
print_table, 41, 45
PROCESS, 4, 33, 39, 41, 47
psych::corr.test(), 10

RECODE, 4, 53
regress, 4, 23, 31, 53
rep_char, 55
RESCALE, 4, 56
RESCALE(), 58
RGB, 56
Run, 4, 57

scaler, 58
set.wd, 4, 58
setwd, 59
show_colors, 4, 59
stats::p.adjust(), 10, 16
STD, 4
STD(%%COMPUTE%%), 64
SUM, 4
SUM(%%COMPUTE%%), 64
texreg::htmlreg(), 40
texreg::screenreg(), 40

theme_bruce, 4, 60
VAR, 24
vars::VAR(), 25
within.1(bruceR-demodata), 6
within.2(bruceR-demodata), 6
within.3(bruceR-demodata), 6