Package 'rosetta'

November 27, 2019

Title Parallel Use of Statistical Packages in Teaching

Version 0.2.0

Date 2019-11-28

Description When teaching statistics, it can often be desirable to uncouple the content from specific software packages. To ease such efforts, the Rosetta Stats website (<https://rosettastats.com>) allows comparing analyses in different packages. This package is the companion to the Rosetta Stats website, aiming to provide functions that produce output that is similar to output from other statistical packages, thereby facilitating 'software-agnostic' teaching of statistics.

Maintainer Gjalt-Jorn Peters <gjalt-jorn@userfriendlyscience.com>

License GPL (>= 3)

URL https://r-packages.gitlab.io/rosetta

BugReports https://gitlab.com/r-packages/rosetta/issues

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Depends R (>= 3.0.0)

Suggests jmvcore (>= 1.0)

Imports car (>= 3.0.2), ggplot2 (>= 2.2.1), ggrepel (>= 0.8), gridExtra (>= 2.3), methods (>= 3.0.0), lavaan (>= 0.6-5), lme4 (>= 1.1.19), multcompView (>= 0.1-0), pander (>= 0.6.3), plyr (>= 1.8.4), psych (>= 1.8.4), pwr (>= 1.2.2), rio (>= 0.5.10), ufs (>= 0.3.0)

NeedsCompilation no

Author Gjalt-Jorn Peters [aut, cre] (<https://orcid.org/0000-0002-0336-9589>), Ron Pat-El [ctb] (<https://orcid.org/0000-0002-4742-0163>), Peter Verboon [ctb] (<https://orcid.org/0000-0001-8656-0890>)

Repository CRAN

Date/Publication 2019-11-27 13:50:06 UTC

R topics documented:

`buildModMedSemModel` *Builds model for moderated mediation anaysis using SEM*

Description

Builds model for moderated mediation anaysis using SEM

Usage

```
buildModMedSemModel(xvar, mvars, yvar, xmmod = NULL, mymod = NULL,
  cmvars = NULL, cyvars = NULL)
```

Arguments

`xvar`	independent variable (predictor)
`mvars`	vector of names of mediators
`yvar`	dependent variable
`xmmod`	moderator of a path(s)
`mymod`	moderator of b path(s)
`cmvars`	covariates for predicting the mediators
`cyvars`	covariates for predicting the dependent variable

Value

lavaan model to be used in moderatedMediationSem

Examples

```
model <-  buildModMedSemModel(xvar="procJustice", mvars= c("cynicism"),
          yvar = "CPB", xmmod = "insecure",mymod = "gender" ,cmvars =c("age"))
```

cpbExample *A test dataset*

Description

The data are about the attitudes of employees of an organisation that is in the middle of a reorganization. The model predicts that feelings of procedural injustice may lead to cynicism and less trust in the management. This relation may be stronger among employees who are insecure about their job continuation. Cynisicm may lead to contra-productive behaviour (CPB). However, strong personal norms may prevent CPB. Cynicism is expected to increase with age, and men may be more inclined towards CPB than women.

Usage

```
cpbExample
```

Format

A data frame with 320 rows and 8 variables:

gender gender participant

age age participant

procJustice prodedural justice

trust trust in management

cynicism cynicism about the management

CPB contr-productive behaviour

insecure insecure about job continuation

norms personal norms about CPB

crossTab *Cross tables*

Description

This function produces a cross table, computes Chi Square, and computes the point estimate and confidence interval for Cramer's V.

Usage

```
crossTab(x, y = NULL, conf.level = 0.95, digits = 2,
  pValueDigits = 3, ...)

## S3 method for class 'crossTab'
print(x, digits = x$input$digits,
  pValueDigits = x$input$pValueDigits, ...)

## S3 method for class 'crossTab'
pander(x, digits = x$input$digits,
  pValueDigits = x$input$pValueDigits, ...)
```

Arguments

`x` Either a crosstable to analyse, or one of two vectors to use to generate that crosstable. The vector should be a factor, i.e. a categorical variable identified as such by the 'factor' class).

`y` If x is a crosstable, y can (and should) be empty. If x is a vector, y must also be a vector.

`conf.level` Level of confidence for the confidence interval.

`digits` Minimum number of digits after the decimal point to show in the result.

`pValueDigits` Minimum number of digits after the decimal point to show in the Chi Square p value in the result.

`...` Extra arguments to `crossTab` are passed on to `ufs::confIntV()`.

Value

The results of `ufs::confIntV()`, but also prints the cross table and the chi square test results.

Examples

```
crossTab(infert$education, infert$induced, samples=50);
```

dlvTheme *dlvPlot*

Description

The dlvPlot function produces a dot-violin-line plot, and dlvTheme is the default theme.

Usage

```
dlvTheme(base_size = 11, base_family = "", ...)

dlvPlot(dat, x = NULL, y, z = NULL, conf.level = 0.95,
  jitter = "FALSE", binnedDots = TRUE, binwidth = NULL,
  error = "lines", dotsize = "density", singleColor = "black",
  comparisonColors = RColorBrewer::brewer.pal(8, "Set1"),
  densityDotBaseSize = 3, normalDotBaseSize = 1, violinAlpha = 0.2,
  dotAlpha = 0.4, lineAlpha = 1, connectingLineAlpha = 1,
  meanDotSize = 5, posDodge = 0.2, errorType = "both",
  outputFile = NULL, outputWidth = 10, outputHeight = 10,
  ggsaveParams = list(units = "cm", dpi = 300, type = "cairo"))

## S3 method for class 'dlvPlot'
print(x, ...)
```

Arguments

`base_size, base_family, ...`
: Passed on to the ggplot theme_grey() function.

`dat`
: The dataframe containing x, y and z.

`x`
: Character value with the name of the predictor ('independent') variable, must refer to a categorical variable (i.e. a factor).

`y`
: Character value with the name of the critetion ('dependent') variable, must refer to a continuous variable (i.e. a numeric vector).

`z`
: Character value with the name of the moderator variable, must refer to a categorical variable (i.e. a factor).

`conf.level`
: Confidence of confidence intervals.

`jitter`
: Logical value (i.e. TRUE or FALSE) whether or not to jitter individual datapoints. Note that jitter cannot be combined with posDodge (see below).

`binnedDots`
: Logical value indicating whether to use binning to display the dots. Overrides jitter and dotsize.

`binwidth`
: Numeric value indicating how broadly to bin (larger values is more binning, i.e. combining more dots into one big dot).

`error`
: Character value: "none", "lines" or "whiskers"; indicates whether to show the confidence interval as lines with (whiskers) or without (lines) horizontal whiskers or not at all (none)

`dotsize` Character value: "density" or "normal"; when "density", the size of each dot corresponds to the density of the distribution at that point.

`singleColor` The color to use when drawing one or more univariate distributions (i.e. when no `z` is specified.

`comparisonColors`
The colors to use when a `z` is specified. This should be at least as many colors as `z` has levels. By default, palette `Set1` from `RColorBrewer` is used.

`densityDotBaseSize`
Numeric value indicating base size of dots when their size corresponds to the density (bigger = larger dots).

`normalDotBaseSize`
Numeric value indicating base size of dots when their size is fixed (bigger = larger dots).

`violinAlpha` Numeric value indicating alpha value of violin layer (0 = completely transparent, 1 = completely opaque).

`dotAlpha` Numeric value indicating alpha value of dot layer (0 = completely transparent, 1 = completely opaque).

`lineAlpha` Numeric value indicating alpha value of the confidence interval line layer (0 = completely transparent, 1 = completely opaque).

`connectingLineAlpha`
Numeric value indicating alpha value of the layer with the lines connecting the means (0 = completely transparent, 1 = completely opaque).

`meanDotSize` Numeric value indicating the size of the dot used to indicate the mean in the line layer.

`posDodge` Numeric value indicating the distance to dodge positions (0 for complete overlap).

`errorType` If the error is shown using lines, this argument indicates Whether the errorbars should show the confidence interval (`errorType='ci'`), the standard errors (`errorType='se'`), or both (`errorType='both'`). In this last case, the standard error will be wider than the confidence interval.

`outputFile` A file to which to save the plot.

`outputWidth, outputHeight`
Width and height of saved plot (specified in centimeters by default, see `ggsaveParams`).

`ggsaveParams` Parameters to pass to ggsave when saving the plot.

Details

This function creates Dot Violin Line plots. One image says more than a thousand words; I suggest you run the example :-)

Value

The behavior of this function depends on the arguments.

If no x and z are provided and y is a character value, dlvPlot produces a univariate plot for the numerical y variable.

If no x and z are provided, and y is c character vector, dlvPlot produces multiple Univariate plots, with variable names determining categories on x-axis and with numerical y variables on y-axis

If both x and y are a character value, and no z is provided, dlvPlot produces a bivariate plot where factor x determines categories on x-axis with numerical variable y on the y-axis (roughly a line plot with a single line)

Finally, if x, y and z are each a character value, dlvPlot produces multivariate plot where factor x determines categories on x-axis, factor z determines the different lines, and with the numerical y variable on the y-axis

An object is returned with the following elements:

`dat.raw`	Raw datafile provided when calling dlvPlot
`dat`	Transformed (long) datafile dlvPlot uses
`descr`	Dataframe with extracted descriptives used to plot the mean and confidence intervals
`yRange`	The range of the Y variable used to construct the plot
`plot`	The plot itself

Examples

```
### Note: the 'not run' is simply because running takes a lot of time,
###       but these examples are all safe to run!
## Not run:
### Create simple dataset
dat <- data.frame(x1 = factor(rep(c(0,1), 20)),
                  x2 = factor(c(rep(0, 20), rep(1, 20))),
                  y=rep(c(4,5), 20) + rnorm(40));
### Generate a simple dlvPlot of y
dlvPlot(dat, y='y');
### Now add a predictor
dlvPlot(dat, x='x1', y='y');
### And finally also a moderator:
dlvPlot(dat, x='x1', y='y', z='x2');
### The number of datapoints might be a bit clearer if we jitter
dlvPlot(dat, x='x1', y='y', z='x2', jitter=TRUE);
### Although just dodging the density-sized dots might work better
dlvPlot(dat, x='x1', y='y', z='x2', posDodge=.3);

## End(Not run)
```

`exportToSPSS` *Basic SPSS translation functions*

Description

Basic functons to make working with R easier for SPSS users: getData and getDat provide an easy way to load SPSS datafiles, and exportToSPSS to write to a datafile and syntax file that SPSS can import; filterBy and useAll allow easy temporary filtering of rows from the dataframe; mediaan and modus compute the median and mode of ordinal or numeric data.

Usage

```
exportToSPSS(dat, savfile = NULL, datafile = NULL, codefile = NULL,
  fileEncoding = "UTF-8", newLinesInString = " |n| ")

filterBy(dat, expression, replaceOriginalDataframe = TRUE,
  envir = parent.frame())

getData(filename = NULL, file = NULL,
  errorMessage = "[defaultErrorMessage]", applyRioLabels = TRUE,
  use.value.labels = FALSE, to.data.frame = TRUE,
  stringsAsFactors = FALSE, silent = FALSE, ...)

getDat(..., dfName = "dat", backup = TRUE)

mediaan(vector)

modus(vector)

useAll(dat, replaceFilteredDataframe = TRUE)
```

Arguments

`dat` Dataframe to process: for filterBy, dataframe to filter rows from; for useAll, dataframe to restore ('unfilter').

`savfile` The name of the SPSS format .sav file (alternative for writing a datafile and a codefile).

`datafile` The name of the data file, a comma separated values file that can be read into SPSS by using the code file.

`codefile` The name of the code file, the SPSS syntax file that can be used to import the data file.

`fileEncoding` The encoding to use to write the files.

`newLinesInString` A string to replace newlines with (SPSS has problems reading newlines).

`expression` Logical expression determining which rows to keep and which to drop. Can be either a logical vector or a string which is then evaluated. If it's a string, it's evaluated using 'with' to evaluate the expression using the variable names.

`replaceOriginalDataframe` Whether to also replace the original dataframe in the parent environment. Very messy, but for maximum compatibility with the 'SPSS way of doing things', by default, this is true. After all, people who care about the messiness/inappropriateness of this function wouldn't be using it in the first place :-)

`envir` The environment where to create the 'backup' of the unfiltered dataframe, for when useAll is called and the filter is deactivated again.

`filename, file` It is possible to specify a path and filename to load here. If not specified, the default R file selection dialogue is shown. `file` is still available for backward compatibility but will eventually be phased out.

`errorMessage` The error message that is shown if the file does not exist or does not have the right extension; "[defaultErrorMessage]" is replaced with a default error message (and can be included in longer messages).

`applyRioLabels` Whether to apply the labels supplied by Rio. This will make variables that has value labels into factors.

`use.value.labels`
Only useful when reading from SPSS files: whether to read variables with value labels as factors (TRUE) or numeric vectors (FALSE).

`to.data.frame` Only useful when reading from SPSS files: whether to return a dataframe or not.

`stringsAsFactors`
Whether to read strings as strings (FALSE) or factors (TRUE).

`silent` Whether to suppress potentially useful information.

`...` Additional options, passed on to the function used to import the data (which depends on the extension of the file).

`dfName` The name of the dataframe to create in the parent environment.

`backup` Whether to backup an object with name `dfName`, if one already exists in the parent environment.

`vector` For mediaan and modus, the vector for which to find the median or mode.

`replaceFilteredDataframe`
Whether to replace the filtered dataframe passed in the 'dat' argument (see replaceOriginalDataframe).

Value

getData returns the imported dataframe, with the filename from which it was read stored in the 'filename' attribute.

getDat is a simple wrapper for `getData()` which creates a dataframe in the parent environment, by default with the name 'dat'. Therefore, calling `getDat()` in the console will allow the user to select a file, and the data from the file will then be read and be available as 'dat'. If an object with `dfName` (i.e. 'dat' by default) already exists, it will be backed up with a warning. `getDat()` therefore returns nothing.

mediaan returns the median, or, in the case of a factor where the median is in between two categories, both categories.

modus returns the mode.

Note

getData() currently can’t read from LibreOffice or OpenOffice files. There doesn’t seem to be a platform-independent package that allows this. Non-CRAN package ROpenOffice from Omega-Hat should be able to do the trick, but fails to install (manual download and installation using http://www.omegahat.org produces "ERROR: dependency ’Rcompression’ is not available for package ’ROpenOffice’" - and manual download and installation of RCompression produces "Please define LIB_ZLIB; ERROR: configuration failed for package ’Rcompression’"). If you have any suggestions, please let me know!

Examples

```
## Not run:
### Open a dialogue to read an SPSS file
getData();

## End(Not run)

### Get a median and a mode
mediaan(c(1,2,2,3,4,4,5,6,6,6,7));
modus(c(1,2,2,3,4,4,5,6,6,6,7));

### Create an example dataframe
(exampleDat <- data.frame(x=rep(8, 8), y=rep(c(0,1), each=4)));
### Filter it, replacing the original dataframe
(filterBy(exampleDat, ”y=0”));
### Restore the old dataframe
(useAll(exampleDat));
```

---

fanova *Flexible anova*

Description

This function is meant as a userfriendly wrapper to approximate the way analysis of variance is done in SPSS.

Usage

```
fanova(data, y, between = NULL, covar = NULL, plot = FALSE,
  levene = FALSE, digits = 2, contrast = NULL)

## S3 method for class 'fanova'
print(x, digits = x$input$digits, ...)
```

Arguments

`data` The dataset containing the variables to analyse.

`y` The dependent variable. For oneway anova, factorial anova, or ancova, this is the name of a variable in dataframe `data`. For repeated measures anova, this is a vector with the names of all variable names in dataframe `data`, e.g. `c('t0_value','t1_value','t2_value')`.

`between` A vector with the variables name(s) of the between subjects factor(s).

`covar` A vector with the variables name(s) of the covariate(s).

`plot` Whether to produce a plot. Note that a plot is only produced for oneway and twoway anova and oneway repeated measures designs: if covariates or more than two between-subjects factors are specified, not plot is produced. For twoway anova designs, the second predictor is plotted as moderator (and the first predictor is plotted on the x axis).

`levene` Whether to show Levene's test for equality of variances (using `car`'s `leveneTest` function but specifying `mean` as function to compute the center of each group).

`digits` Number of digits (actually: decimals) to use when printing results. The p-value is printed with one extra digit.

`contrast` This functionality has not been implemented yet.

`x` The object to print (i.e. as produced by `regr`).

`...` Any additional arguments are ignored.

Details

This wrapper uses `oneway` and `lm` and `lmer` in combination with `car`'s `Anova` function to conduct the analysis of variance.

Value

Mainly, this function prints its results, but it also returns them in an object containing three lists:

`input` The arguments specified when calling the function

`intermediate` Intermediat objects and values

`output` The results such as the plot.

Author(s)

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

`regr` and `logRegr` for similar functions for linear and logistic regression and `oneway`, `lm`, `lmer` and `Anova` for the functions used behind the scenes.

Examples

```
### Oneway anova with a plot
fanova(dat=mtcars, y='mpg', between='cyl', plot=TRUE);

### Factorial anova
fanova(dat=mtcars, y='mpg', between=c('vs', 'am'), plot=TRUE);

### Ancova
fanova(dat=mtcars, y='mpg', between=c('vs', 'am'), covar='hp');

### Don't run these examples to not take too much time during testing
### for CRAN
## Not run:
### Repeated measures anova; first generate datafile
dat <- mtcars[, c('am', 'drat', 'wt')];
names(dat) <- c('factor', 't0_dependentVar' ,'t1_dependentVar');
dat$factor <- factor(dat$factor);

### Then do the repeated measures anova
fanova(dat, y=c('t0_dependentVar' ,'t1_dependentVar'),
       between='factor', plot=TRUE);

## End(Not run)
```

freq *Frequency tables*

Description

Function to show frequencies in a manner similar to what SPSS' "FREQUENCIES" command does. Note that `frequency` is an alias for `freq`.

Usage

```
freq(vector, digits = 1, nsmall = 1, transposed = FALSE, round = 1,
  plot = FALSE, plotTheme = ggplot2::theme_bw())

## S3 method for class 'freq'
print(x, digits = x$input$digits,
  nsmall = x$input$nsmall, transposed = x$input$transposed, ...)

## S3 method for class 'freq'
pander(x, ...)

frequencies(..., digits = 1, nsmall = 1, transposed = FALSE,
  round = 1, plot = FALSE, plotTheme = ggplot2::theme_bw())
```

```
## S3 method for class 'frequencies'
print(x, ...)

## S3 method for class 'frequencies'
pander(x, prefix = "###", ...)
```

Arguments

vector	A vector of values to compute frequencies for.
digits	Minimum number of significant digits to show in result.
nsmall	Minimum number of digits after the decimal point to show in the result.
transposed	Whether to transpose the results when printing them (this can be useful for blind users).
round	Number of digits to round the results to (can be used in conjunction with digits to determine format of results).
plot	If true, a histogram is shown of the variable.
plotTheme	The ggplot2 theme to use.
x	The `freq` or `frequencies` object to print.
...	For `frequencies`, the variables of which to provide frequencies; for the `print` methods, additional arguments are passed on to the `print` function.
prefix	The prefix to use when printing `frequencies`, to easily prepend Markdown headers.

Value

An object with several elements, the most notable of which is:

dat	A dataframe with the frequencies

For `frequencies`, these objects are in a list of their own.

Examples

```
### Create factor vector
ourFactor <- factor(mtcars$gear, levels=c(3,4,5),
                    labels=c("three", "four", "five"));
### Add some missing values
factorWithMissings <- ourFactor;
factorWithMissings[10] <- factorWithMissings[20] <- NA;

### Show frequencies
freq(ourFactor);
freq(factorWithMissings);

### ... Or for all of them at one
```

```
frequencies(ourFactor, factorWithMissings);
```

`freqjmv` *Frequencies*

Description

Frequencies

Usage

```
freqjmv(data, vector)
```

Arguments

`data`	.
`vector`	.

Value

A results object containing:

`results$text`	a preformatted

`gemm` *Analyze moderated mediation model using SEM*

Description

Analyze moderated mediation model using SEM

Usage

```
gemm(data = NULL, xvar, mvars, yvar, xmmod = NULL, mymod = NULL,
  cmvars = NULL, cyvars = NULL, estMethod = "bootstrap",
  nboot = 1000)
```

Arguments

`data`	data frame
`xvar`	predictor variable, must be either numerical or dichotomous
`mvars`	vector of names of mediator variables

`yvar` dependent variable, must be numerical

`xmmod` moderator of effect predictor on mediators, must be either numerical or dichotomous

`mymod` moderator of effect mediators on dependent variable, must be either numerical or dichotomous

`cmvars` covariates for mediators

`cyvars` covariates for dependent variable

`estMethod` estimation of standard errors method, bootstrap is default

`nboot` number of bootstrap samples

Value

gemm object

Examples

```
## Not run:
data("cpbExample")
       res <- gemm(dat = cpbExample, xvar="procJustice", mvars= c("cynicism","trust"),
       yvar = "CPB", nboot=500)
print(res)

## End(Not run)
```

`ggBoxplot` *Box plot using ggplot*

Description

This function provides a simple interface to create a `ggplot` box plot, organising different boxplots by levels of a factor is desired, and showing row numbers of outliers.

Usage

```
ggBoxplot(dat, y = NULL, x = NULL, labelOutliers = TRUE,
  outlierColor = "red", theme = ggplot2::theme_bw(), ...)
```

Arguments

`dat` Either a vector of values (to display in the box plot) or a dataframe containing variables to display in the box plot.

`y` If `dat` is a dataframe, this is the name of the variable to make the box plot of.

`x` If `dat` is a dataframe, this is the name of the variable (normally a factor) to place on the X axis. Separate box plots will be generate for each level of this variable.

`labelOutliers` Whether or not to label outliers.

`outlierColor` If labeling outliers, this is the color to use.

`theme` The theme to use for the box plot.

`...` Any additional arguments will be passed to `geom_boxplot`.

Details

This function is based on JasonAizkalns' answer to a question on Stack Exchange (Cross Validated; see `http://stackoverflow.com/questions/33524669/labeling-outliers-of-boxplots-in-r`).

Value

A `ggplot` plot is returned.

Author(s)

Jason Aizkalns; implemented in this package (and tweaked a bit) by Gjalt-Jorn Peters.

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

`geom_boxplot`

Examples

```
### A box plot for miles per gallon in the mtcars dataset:
ggBoxplot(mtcars$mpg);

### And separate for each level of 'cyl' (number of cylinder):
ggBoxplot(mtcars, y='mpg', x='cyl');
```

---

logRegr *Userfriendly wrapper to do logistic regression in R*

Description

This function is meant as a userfriendly wrapper to approximate the way logistic regression is done in SPSS.

Usage

```
logRegr(formula, data = NULL, conf.level = 0.95, digits = 2,
  pvalueDigits = 3, crossTabs = TRUE, plot = FALSE,
  collinearity = FALSE, env = parent.frame(),
  predictionColor = viridis::viridis(3)[3], predictionAlpha = 0.5,
  predictionSize = 2, dataColor = viridis::viridis(3)[1],
  dataAlpha = 0.33, dataSize = 2,
  observedMeansColor = viridis::viridis(3)[2], binObservedMeans = 7,
  observedMeansSize = 2, observedMeansWidth = NULL,
  observedMeansAlpha = 0.5, theme = ggplot2::theme_bw())
```

```
## S3 method for class 'logRegr'
print(x, digits = x$input$digits,
  pvalueDigits = x$input$pvalueDigits, ...)
```

Arguments

`formula` The formula, specified in the same way as for `stats::glm()` (which is used for the actual analysis).

`data` Optionally, a dataset containing the variables in the formula (if not specified, the variables must exist in the environment specified in `env`.

`conf.level` The confidence level for the confidence intervals.

`digits` The number of digits used when printing the results.

`pvalueDigits` The number of digits used when printing the p-values.

`crossTabs` Whether to show cross tabulations of the correct predictions for the null model and the tested model, as well as the percentage of correct predictions.

`plot` Whether to display the plot.

`collinearity` Whether to show collinearity diagnostics.

`env` If no dataframe is specified in `data`, use this argument to specify the environment holding the variables in the formula.

`predictionColor, dataColor, observedMeansColor`
The color of, respectively, the line and confidence interval showing the prediction; the points representing the observed data points; and the means based on the observed data.

`predictionAlpha, dataAlpha, observedMeansAlpha`
The alpha of, respectively, the confidence interval of the prediction; the points representing the observed data points; and the means based on the observed data (set to 0 to hide an element).

`predictionSize, dataSize, observedMeansSize`
The size of, respectively, the line of the prediction; the points representing the observed data points; and the means based on the observed data (set to 0 to hide an element).

`binObservedMeans`
Whether to bin the observed means; either FALSE or a single numeric value specifying the number of bins.

`observedMeansWidth`
The width of the lines of the observed means. If not specified (i.e. `NULL`), this is computed automatically and set to the length of the shortest interval between two successive points in the predictor data series (found using `ufs::findShortestInterval()`.

`theme` The theme used to display the plot.

`x` The object to print.

`...` Any additional arguments are passed to the default print method.

Details

This function

Value

Mainly, this function prints its results, but it also returns them in an object containing three lists:

`input`	The arguments specified when calling the function
`intermediate`	Intermediat objects and values
`output`	The results, such as the plot, the cross tables, and the coefficients.

Author(s)

Ron Pat-El & Gjalt-Jorn Peters (both while at the Open University of the Netherlands)

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

`regr` and `fanova` for similar functions for linear regression and analysis of variance and `stats::glm()` for the regular interface for logistic regression.

Examples

```
### Simplest way to call logRegr
logRegr(data=mtcars, formula = vs ~ mpg);

### Also ordering a plot
logRegr(data=mtcars, formula = vs ~ mpg, plot=TRUE);

### Only use five bins
logRegr(data=mtcars, formula = vs ~ mpg, plot=TRUE, binObservedMeans=5);
```

`meanDiff` *meanDiff*

Description

The meanDiff function compares the means between two groups. It computes Cohen's d, the unbiased estimate of Cohen's d (Hedges' g), and performs a t-test. It also shows the achieved power, and, more usefully, the power to detect small, medium, and large effects.

Usage

```
meanDiff(x, y = NULL, paired = FALSE, r.prepost = NULL,
  var.equal = "test", conf.level = 0.95, plot = FALSE, digits = 2,
  envir = parent.frame())

## S3 method for class 'meanDiff'
print(x, digits = x$digits, powerDigits = x$digits +
  2, ...)

## S3 method for class 'meanDiff'
pander(x, digits = x$digits, powerDigits = x$digits
  + 2, ...)
```

Arguments

`x` Dichotomous factor: variable 1; can also be a formula of the form y ~ x, where x must be a factor with two levels (i.e. dichotomous).

`y` Numeric vector: variable 2; can be empty if x is a formula.

`paired` Boolean; are x & y independent or dependent? Note that if x & y are dependent, they need to have the same length.

`r.prepost` Correlation between the pre- and post-test in the case of a paired samples t-test. This is required to compute Cohen's d using the formula on page 29 of Borenstein et al. (2009). If NULL, the correlation is simply computed from the provided scores (but of course it will then be lower if these is an effect - this will lead to an underestimate of the within-groups variance, and therefore, of the standard error of Cohen's d, and therefore, to confidence intervals that are too narrow (too liberal). Also, of course, when using this data to compute the within-groups correlation, random variations will also impact that correlation, which means that confidence intervals may in practice deviate from the null hypothesis significance testing p-value in either direction (i.e. the p-value may indicate a significant association while the confidence interval contains 0, or the other way around). Therefore, if the test-retest correlation of the relevant measure is known, please provide this here to enable computation of accurate confidence intervals.

`var.equal` String; only relevant if x & y are independent; can be "test" (default; test whether x & y have different variances), "no" (assume x & y have different variances; see the Warning below!), or "yes" (assume x & y have the same variance)

`conf.level` Confidence of confidence intervals you want.

`plot` Whether to print a dlvPlot.

`digits` With what precision you want the results to print.

`envir` The environment where to search for the variables (useful when calling meanDiff from a function where the vectors are defined in that functions environment).

`powerDigits` With what precision you want the power to print.

`...` Additional arguments are passen on to the `ggplot2::ggplot()` print method.

Details

This function uses the formulae from Borenstein, Hedges, Higgins & Rothstein (2009) (pages 25-32).

Value

An object is returned with the following elements:

`variables`	Input variables
`groups`	Levels of the x variable, the dichotomous factor
`ci.confidence`	Confidence of confidence intervals
`digits`	Number of digits for output
`x`	Values of dependent variable in first group
`y`	Values of dependent variable in second group
`type`	Type of t-test (independent or dependent, equal variances or not)
`n`	Sample sizes of the two groups
`mean`	Means of the two groups
`sd`	Standard deviations of the two groups
`objects`	Objects used; the t-test and optionally the test for equal variances
`variance`	Variance of the difference score
`meanDiff`	Difference between the means
`meanDiff.d`	Cohen's d
`meanDiff.d.var`	Variance of Cohen's d
`meanDiff.d.se`	Standard error of Cohen's d
`meanDiff.J`	Correction for Cohen's d to get to the unbiased Hedges g
`power`	Achieved power with current effect size and sample size
`power.small`	Power to detect small effects with current sample size
`power.medium`	Power to detect medium effects with current sample size
`power.largel`	Power to detect large effects with current sample size
`meanDiff.g`	Hedges' g
`meanDiff.g.var`	Variance of Hedges' g
`meanDiff.g.se`	Standard error of Hedges' g
`ci.usedZ`	Z value used to compute confidence intervals
`meanDiff.d.ci.lower`	Lower bound of confidence interval around Cohen's d
`meanDiff.d.ci.upper`	Upper bound of confidence interval around Cohen's d
`meanDiff.g.ci.lower`	Lower bound of confidence interval around Hedges' g
`meanDiff.g.ci.upper`	Upper bound of confidence interval around Hedges' g
`meanDiff.ci.lower`	Lower bound of confidence interval around raw mean
`meanDiff.ci.upper`	Upper bound of confidence interval around raw mean
`t`	Student t value for Null Hypothesis Significance Testing
`df`	Degrees of freedom for t value
`p`	p-value corresponding to t value

Warning

Note that when different variances are assumed for the t-test (i.e. the null-hypothesis test), the values of Cohen's d are still based on the assumption that the variance is equal. In this case, the confidence interval might, for example, not contain zero even though the NHST has a non-significant p-value (the reverse can probably happen, too).

References

Borenstein, M., Hedges, L. V., Higgins, J. P., & Rothstein, H. R. (2011). Introduction to meta-analysis. John Wiley & Sons.

Examples

```
### Create simple dataset
dat <- PlantGrowth[1:20,];
### Remove third level from group factor
dat$group <- factor(dat$group);
### Compute mean difference and show it
meanDiff(dat$weight ~ dat$group);

### Look at second treatment
dat <- rbind(PlantGrowth[1:10,], PlantGrowth[21:30,]);
### Remove third level from group factor
dat$group <- factor(dat$group);
### Compute mean difference and show it
meanDiff(x=dat$group, y=dat$weight);
```

`meanDiff.multi` *meanDiff.multi*

Description

The meanDiff.multi function compares many means for many groups. It presents the results in a dataframe summarizing all relevant information, and produces plot showing the confidence intervals for the effect sizes for each predictor (i.e. dichotomous variable). Like meanDiff, it computes Cohen's d, the unbiased estimate of Cohen's d (Hedges' g), and performs a t-test. It also shows the achieved power, and, more usefully, the power to detect small, medium, and large effects.

Usage

```
meanDiff.multi(dat, y, x = NULL, var.equal = "yes",
  conf.level = 0.95, digits = 2, orientation = "vertical",
  zeroLineColor = "grey", zeroLineSize = 1.2, envir = parent.frame())

## S3 method for class 'meanDiff.multi'
print(x, digits = x$digits,
  powerDigits = x$digits + 2, ...)
```

Arguments

`dat` The dataframe containing the variables involved in the mean tests.

`y` Character vector containing the list of interval variables to include in the tests.

`x` Character vector containing the list of the dichotomous variables to include in the tests. If x is empty, paired samples t-tests will be conducted.

`var.equal` String; only relevant if x & y are independent; can be "test" (default; test whether x & y have different variances), "no" (assume x & y have different variances; see the Warning below!), or "yes" (assume x & y have the same variance)

`conf.level` Confidence of confidence intervals you want.

`digits` With what precision you want the results to print.

`orientation` Whether to plot the effect size confidence intervals vertically (like a forest plot, the default) or horizontally.

`zeroLineColor` Color of the horizontal line at an effect size of 0 (set to 'white' to not display the line; also adjust the size to 0 then).

`zeroLineSize` Size of the horizontal line at an effect size of 0 (set to 0 to not display the line; also adjust the color to 'white' then).

`envir` The environment where to search for the variables (useful when calling meanDiff from a function where the vectors are defined in that functions environment).

`powerDigits` With what precision you want the power to print.

`...` Additional arguments are passed on to the `meanDiff()` print methods.

Details

This function uses the meanDiff function, which uses the formulae from Borenstein, Hedges, Higgins & Rothstein (2009) (pages 25-32).

Value

An object is returned with the following elements:

`results.raw` Objects returned by the calls to meanDiff.

`plots` For every comparison, a plot with the datapoints, means, and confidence intervals in the two groups.

`results.compiled` Dataframe with the most important results from each comparison.

`plots.compiled` For every dichotomous (x) variable, a plot with the confidence interval for the effect size of each dependent (y) variable.

`input` The arguments with which the function was called.

Warning

Note that when different variances are assumed for the t-test (i.e. the null-hypothesis test), the values of Cohen's d are still based on the assumption that the variance is equal. In this case, the confidence interval might, for example, not contain zero even though the NHST has a non-significant p-value (the reverse can probably happen, too).

References

Borenstein, M., Hedges, L. V., Higgins, J. P., & Rothstein, H. R. (2011). Introduction to meta-analysis. John Wiley & Sons.

Examples

```
### Create simple dataset
dat <- data.frame(x1 = factor(rep(c(0,1), 20)),
                  x2 = factor(c(rep(0, 20), rep(1, 20))),
                  y=rep(c(4,5), 20) + rnorm(40));
### Compute mean difference and show it
meanDiff.multi(dat, x=c('x1', 'x2'), y='y', var.equal="yes");
```

---

oneway *oneway*

Description

The oneway function wraps a number of analysis of variance functions into one convenient interface that is similar to the oneway anova command in SPSS.

Usage

```
oneway(y, x, posthoc = NULL, means = FALSE, fullDescribe = FALSE,
  levene = FALSE, plot = FALSE, digits = 2, omegasq = TRUE,
  etasq = TRUE, corrections = FALSE, pvalueDigits = 3, t = FALSE,
  conf.level = 0.95, posthocLetters = FALSE,
  posthocLetterAlpha = 0.05, overrideVarNames = NULL, silent = FALSE)

## S3 method for class 'oneway'
print(x, digits = x$input$digits,
  pvalueDigits = x$input$pvalueDigits, na.print = "", ...)

## S3 method for class 'oneway'
pander(x, digits = x$input$digits,
  pvalueDigits = x$input$pvalueDigits, headerStyle = "**",
  na.print = "", ...)
```

Arguments

`y`	y has to be a numeric vector.
`x`	x has to be vector that either is a factor or can be converted into one.
`posthoc`	Which post-hoc tests to conduct. Valid values are any correction methods in p.adjust.methods (at the time of writing of this document, "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"), as well as "tukey" and "games-howell".

`means` Whether to show the means for the y variable in each of the groups determined by the x variable.

`fullDescribe` If TRUE, not only the means are shown, but all statistics acquired through the 'describe' function in the 'psych' package are shown.

`levene` Whether to show Levene's test for equality of variances (using `car`'s `leveneTest` function but specifying `mean` as function to compute the center of each group).

`plot` Whether to show a plot of the means of the y variable in each of the groups determined by the x variable.

`digits` The number of digits to show in the output.

`omegasq` Whether to show the omega squared effect size.

`etasq` Whether to show the eta squared effect size (this is biased and generally advised against; omega squared is less biased).

`corrections` Whether to show the corrections for unequal variances (Welch and Brown-Forsythe).

`pvalueDigits` The number of digits to show for p-values; smaller p-values will be shown as <.001 or <.0001 etc.

`t` Whether to transpose the dataframes with the means (if requested) and the anova results. This can be useful for blind people.

`conf.level` Confidence level to use when computing the confidence interval for eta^2. Note that the function we use doubles the 'unconfidence' level to maintain consistency with the NHST value (see http://yatani.jp/HCIstats/ANOVA#RCodeOneWay, http://daniellakens.blogspot.nl/2014/06/calculating-confidence-intervals-for.html or Steiger, J. H. (2004). Beyond the F test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. Psychological methods, 9(2), 164-82. doi:10.1037/1082-989X.9.2.164

`posthocLetters` Whether to also compute and show the letters signifying differences between groups when conducting post hoc tests. This requires package `multcompView` to be installed.

`posthocLetterAlpha`
The alpha to use when determining whether groups have different means when using `posthocLetters`.

`overrideVarNames`
Can be used to override the variable names (most useful in functions).

`silent` Whether to show warnings and other diagnostic information or remain silent.

`na.print` How to print missing values.

`...` Any additional arguments are passed to the `print` or `pander` function.

`headerStyle` The header pre- and suffix to use when pandering the result (useful when working with Markdown).

Value

A list of three elements:

`input` List with input arguments

intermediate List of intermediate objects, such as the aov and Anova (from the car package) objects.

output List with etasq, the effect size, and dat, a dataframe with the Oneway Anova results.

Note

By my knowledge the Brown-Forsythe correction was not yet available in R. I took this from the original paper (directed there by Field, 2014). Note that this is the corrected *F* value, not the Brown-Forsythe test for normality!

Author(s)

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

References

Brown, M., & Forsythe, A. (1974). *The small sample behavior of some statistics which test the equality of several means.* Technometrics, 16(1), 129-132. https://doi.org/10.2307/1267501

Field, A. (2014) *Discovering statistics using SPSS* (4th ed.). London: Sage.

Steiger, J. H. (2004). *Beyond the F test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis*. Psychological methods, 9(2), 164-82. doi:10.1037/1082-989X.9.2.164

Examples

```
### Do a oneway Anova
oneway(y=ChickWeight$weight, x=ChickWeight$Diet);

### Also order means and transpose the results
oneway(y=ChickWeight$weight, x=ChickWeight$Diet, means=TRUE, t=TRUE);
```

plotIMM *Makes plot of Index of Moderated Mediation of gemm object*

Description

Makes plot of Index of Moderated Mediation of gemm object

Usage

```
plotIMM(x, ...)
```

Arguments

`x`	object moderatedMediationSem
`...`	optional

Value

simple slope plots for each mediator and simple slopes parameter estimates

plotSS *Makes simple slope plots of gemm object*

Description

Makes simple slope plots of gemm object

Usage

```
plotSS(x, ...)
```

Arguments

`x`	object moderatedMediationSem
`...`	optional

Value

simple slope plots for each mediator and simple slopes parameter estimates

posthocTGH *posthocTGH*

Description

This function is used by the 'oneway' function for oneway analysis of variance in case a user requests post-hoc tests using the Tukey or Games-Howell methods.

Usage

```
posthocTGH(y, x, method = c("games-howell", "tukey"),
  conf.level = 0.95, digits = 2, p.adjust = "none",
  formatPvalue = TRUE)

## S3 method for class 'posthocTGH'
print(x, digits = x$input$digits, ...)
```

Arguments

y	y has to be a numeric vector.
x	x has to be vector that either is a factor or can be converted into one.
method	Which post-hoc tests to conduct. Valid values are "tukey" and "games-howell".
conf.level	Confidence level of the confidence intervals.
digits	The number of digits to show in the output.
p.adjust	Any valid p.adjust method.
formatPvalue	Whether to format the p values according to APA standards (i.e. replace all values lower than .001 with '<.001'). This only applies to the printing of the object, not to the way the p values are stored in the object.
...	Any additional arguments are passed on to the print function.

Value

A list of three elements:

input	List with input arguments
intermediate	List of intermediate objects.
output	List with two objects 'tukey' and 'games.howell', containing the outcomes for the respective post-hoc tests.

Note

This function is based on a file that was once hosted at http://www.psych.yorku.ca/cribbie/6130/games_howell.R, but has been removed since. It was then adjusted for implementation in the userfriendlyscience::userfriendlyscience package. Jeffrey Baggett needed the confidence intervals, and so emailed them, after which his updated function was used. In the meantime, it appears Aaron Schlegel (https://rpubs.com/aaronsc32) independently developed a version with confidence intervals and posted it on RPubs at https://rpubs.com/aaronsc32/games-howell-test.

Also, for some reason, p.adjust can be used to specify additional correction of *p* values. I'm not sure why I implemented this, but I'm not entirely sure it was a mistake either. Therefore, in userfriendlyscience version 0.6-2, the default of this setting changed from "holm" to "none" (also see https://stats.stackexchange.com/questions/83941/games-howell-post-hoc-test-in-r).

Author(s)

Gjalt-Jorn Peters (Open University of the Netherlands) & Jeff Bagget (University of Wisconsin - La Crosse)

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

```
### Compute post-hoc statistics using the tukey method
posthocTGH(y=ChickWeight$weight, x=ChickWeight$Diet, method="tukey");
### Compute post-hoc statistics using the games-howell method
posthocTGH(y=ChickWeight$weight, x=ChickWeight$Diet);
```

prepPlotIMM *Makes Index of Mediated Moderated plots*

Description

Makes Index of Mediated Moderated plots

Usage

```
prepPlotIMM(data, xvar, yvar, mod, mvars, parEst, vdichotomous, modLevels,
  path = NULL)
```

Arguments

`data`	data frame containg the variables of the model
`xvar`	predictor variable name
`yvar`	depedendent variable name
`mod`	moderator name
`mvars`	vector of mediators names
`parEst`	parameter estimates from lavaan results
`vdichotomous`	indicates whether moderator is dichotomous (TRUE)
`modLevels`	levels of dichotomous moderator
`path`	which path is used

Value

empty, directly plots all simple slopes and all indices of mediation

prepPlotSS *Makes simple slope plots*

Description

Makes simple slope plots

Usage

```
prepPlotSS(data, xvar, yvar, mod, mvars, parEst, vdichotomous, modLevels,
  predLevels = NULL, xquant, yquant, path = NULL)
```

Arguments

`data`	data frame containg the variables of the model
`xvar`	predictor variable name
`yvar`	depedendent variable name
`mod`	moderator name
`mvars`	vector of mediators names
`parEst`	parameter estimates from lavaan results
`vdichotomous`	indicates whether moderator is dichotomous (TRUE)
`modLevels`	levels of dichotomous moderator
`predLevels`	levels of dichotomous moderator
`xquant`	quantiles of x
`yquant`	quantiles of y
`path`	which path is used

Value

empty, directly plots all simple slopes and all indices of mediation

`print.gemm` *print method of object of class gemm*

Description

print method of object of class gemm

Usage

```
## S3 method for class 'gemm'
print(x, ..., digits = 2, silence = FALSE)
```

Arguments

`x`	object of class gemm
`...`	additional pars
`digits`	number of digits
`silence`	boolean, if true out is not printed

`regr` *regr: a simple regression analysis wrapper*

Description

The `regr` function wraps a number of linear regression functions into one convenient interface that provides similar output to the regression function in SPSS. It automatically provides confidence intervals and standardized coefficients. Note that this function is meant for teaching purposes, and therefore it's only for very basic regression analyses.

Usage

```
regr(formula, data = NULL, conf.level = 0.95, digits = 2,
  pvalueDigits = 3, coefficients = c("raw", "scaled"), plot = FALSE,
  pointAlpha = 0.5, collinearity = FALSE, influential = FALSE,
  ci.method = c("widest", "r.con", "olkinfinn"),
  ci.method.note = FALSE, env = parent.frame())

## S3 method for class 'regr'
print(x, digits = x$input$digits,
  pvalueDigits = x$input$pvalueDigits, ...)

## S3 method for class 'regr'
pander(x, digits = x$input$digits,
  pvalueDigits = x$input$pvalueDigits, ...)
```

Arguments

`formula` The formula of the regression analysis, of the form `y ~ x1 + x2`, where y is the dependent variable and x1 and x2 are the predictors.

`data` If the terms in the formula aren't vectors but variable names, this should be the dataframe where those variables are stored.

`conf.level` The confidence of the confidence interval around the regression coefficients.

`digits` Number of digits to round the output to.

`pvalueDigits` The number of digits to show for p-values; smaller p-values will be shown as <.001 or <.0001 etc.

`coefficients` Which coefficients to show; can be "raw" to only show the raw (unstandardized) coefficients; "scaled" to only show the scaled (standardized) coefficients), or c("raw", "scaled') to show both.

`plot` For regression analyses with only one predictor (also sometimes confusingly referred to as 'univariate' regression analyses), scatterplots with regression lines and their standard errors can be produced.

`pointAlpha` The alpha channel (transparency, or rather: 'opaqueness') of the points drawn in the plot.

`collinearity` Whether to compute and show collinearity diagnostics (specifically, the tolerance (*1 - R^2*, where *R^2* is the one obtained when regressing each predictor on all the other predictors) and the Variance Inflation Factor (VIF), which is the reciprocal of the tolerance, i.e. *VIF = 1 / tolerance*).

`influential` Whether to compute diagnostics for influential cases. These are stored in the returned object in the `lm.influence.raw` and `lm.influence.scaled` objects in the `intermediate` object.

`ci.method, ci.method.note`
Which method to use for the confidence interval around R squared, and whether to display a note about this choice.

`env` The enviroment where to evaluate the formula.

`x` The object to print (i.e. as produced by `regr`).

`...` Any additional arguments are ignored.

Value

A list of three elements:

`input` List with input arguments

`intermediate` List of intermediate objects, such as the lm and confint objects.

`output` List with two dataframes, one with the raw coefficients, and one with the scaled coefficients.

Author(s)

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

```
### Do a simple regression analysis
regr(age ~ circumference, dat=Orange);

### Show more digits for the p-value
regr(Orange$age ~ Orange$circumference, pvalueDigits=18);
```

Index