Package 'rosetta'

November 24, 2020

Title Parallel Use of Statistical Packages in Teaching

Version 0.3.1

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 7.1.1

Depends R (>= 3.0.0)

Suggests GPArotation, jmvcore (>= 1.0), MBESS (>= 4.8), rmarkdown, testthat

Imports car (>= 3.0.2), diptest (>= 0.75), ggplot2 (>= 2.2), ggrepel (>= 0.8), gridExtra (>= 2.3), methods (>= 3.0), lavaan (>= 0.6-5), lme4 (>= 1.1.19), multcompView (>= 0.1), pander (>= 0.6.3), plyr (>= 1.8.4), psych (>= 1.8.4), pwr (>= 1.2.2), rio (>= 0.5.10), rmdpartials (>= 0.5.8), ufs (>= 0.4.0), knitr, kableExtra

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>), Melissa Gordon Wolf [ctb] (<https://orcid.org/0000-0001-7677-0659>)

Repository CRAN

Date/Publication 2020-11-24 12:10:05 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"))
```

confIntSD *Confidence interval for standard deviation*

Description

This function is vectorized.

Usage

```
confIntSD(x, n = NULL, conf.level = 0.95)
```

Arguments

x	Either a standard deviation, in which case n must also be provided, or a vector, in which case n must be NULL.
n	The sample size is x is a standard deviation.
conf.level	The confidence level

Value

A vector or matrix.

Examples

```
rosetta::confIntSD(mtcars$mpg);
rosetta::confIntSD(c(6, 7), c(32, 32));
```

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);
```

`descr` *descr (or descriptives)*

Description

This function provides a number of descriptives about your data, similar to what SPSS's DESCRIPTIVES (often called with DESCR) does.

Usage

```
descr(
  x,
  items = names(x),
  varLabels = NULL,
  mean = TRUE,
  meanCI = TRUE,
  median = TRUE,
  mode = TRUE,
  var = TRUE,
  sd = TRUE,
  se = FALSE,
  min = TRUE,
  max = TRUE,
  q1 = FALSE,
  q3 = FALSE,
  IQR = FALSE,
  skewness = TRUE,
  kurtosis = TRUE,
  dip = TRUE,
  totalN = TRUE,
  missingN = TRUE,
  validN = TRUE,
  histogram = FALSE,
  boxplot = FALSE,
  digits = 2,
  errorOnFactor = FALSE,
  convertFactor = FALSE,
  maxModes = 1,
```

```
  maxPlotCols = 4,
  t = FALSE,
  headingLevel = 3,
  conf.level = 0.95,
  quantileType = 2
)

rosettaDescr_partial(
  x,
  digits = attr(x, "digits"),
  show = attr(x, "show"),
  headingLevel = attr(x, "headingLevel"),
  maxPlotCols = attr(x, "maxPlotCols"),
  echoPartial = FALSE,
  partialFile = NULL,
  quiet = TRUE,
  ...
)

## S3 method for class 'rosettaDescr'
knit_print(
  x,
  digits = attr(x, "digits"),
  show = attr(x, "show"),
  headingLevel = attr(x, "headingLevel"),
  maxPlotCols = attr(x, "maxPlotCols"),
  echoPartial = FALSE,
  partialFile = NULL,
  quiet = TRUE,
  ...
)

## S3 method for class 'rosettaDescr'
print(
  x,
  digits = attr(x, "digits"),
  show = attr(x, "show"),
  maxPlotCols = attr(x, "maxPlotCols"),
  headingLevel = attr(x, "headingLevel"),
  forceKnitrOutput = FALSE,
  ...
)
```

Arguments

x	The object to print (i.e. as produced by descr).
items	Optionally, if x is a data frame, the variable names for which to produce the descriptives.

varLabels — Optionally, a named vector with 'pretty labels' to show for the variables. This has to be a vector of the same length as items, and if it is not a named vector with the names corresponding to the items, it has to be in the same order.

mean, meanCI, median, mode — Whether to compute the mean, its confidence interval, the median, and/or the mode (all logical, so TRUE or FALSE).

var, sd, se — Whether to compute the variance, standard deviation, and standard error (all logical, so TRUE or FALSE).

min, max, q1, q3, IQR — Whether to compute the minimum, maximum, first and third quartile, and interquartile range (all logical, so TRUE or FALSE).

skewness, kurtosis, dip — Whether to compute the skewness, kurtosis and dip test (all logical, so TRUE or FALSE).

totalN, missingN, validN — Whether to show the total sample size, the number of missing values, and the number of valid (i.e. non-missing) values (all logical, so TRUE or FALSE).

histogram, boxplot — Whether to show a histogram and/or boxplot

digits — The number of digits to round the results to when showing them.

errorOnFactor, convertFactor — If errorOnFactor is TRUE, factors throw an error. If not, if convertFactor is TRUE, they will be converted to numeric values using as.numeric(as.character(x)), and then the same output will be generated as for numeric variables. If convertFactor is false, the frequency table will be produced.

maxModes — Maximum number of modes to display: displays "multi" if more than this number of modes if found.

maxPlotCols — The maximum number of columns when plotting multiple histograms and/or boxplots.

t — Whether to transpose the dataframes when printing them to the screen (this is easier for users relying on screen readers). **Note: this functionality has not yet been implemented!**

headingLevel — The number of hashes to print in front of the headings when printing while knitting

conf.level — Confidence of confidence interval around the mean in the central tendency measures.

quantileType — The type of quantiles to be used to compute the interquartile range (IQR). See quantile for more information.

show — A vector of elements to show in the results, based on the arguments that activate/deactivate the descriptives (from mean to validN).

echoPartial — Whether to show the executed code in the R Markdown partial (TRUE) or not (FALSE).

partialFile — This can be used to specify a custom partial file. The file will have object x available.

quiet — Passed on to knitr::knit() whether it should b chatty (FALSE) or quiet (TRUE).

... — Any additional arguments are passed to the default print method by the print method, and to rmdpartials::partial() when knitting an RMarkdown partial.

forceKnitrOutput — Force knitr output.

Details

Note that R (of course) has many similar functions, such as summary, psych::describe() in the excellent psych::psych package.

The Hartigans' Dip Test may be unfamiliar to users; it is a measure of uni- vs. multimodality, computed by diptest::dip.test() from the dip.test package. Depending on the sample size, values over .025 can be seen as mildly indicative of multimodality, while values over .05 probably warrant closer inspection (the p-value can be obtained using diptest::dip.test(); also see Table 1 of Hartigan & Hartigan (1985) for an indication as to critical values).

Value

A list of dataframes with the requested values.

Author(s)

Gjalt-Jorn Peters

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

References

Hartigan, J. A.; Hartigan, P. M. The Dip Test of Unimodality. Ann. Statist. 13 (1985), no. 1, 70–84. doi:10.1214/aos/1176346577. https://projecteuclid.org/euclid.aos/1176346577.

See Also

summary, [psych::describe()

Examples

```
### Simplest example with default settings
descr(mtcars$mpg);

### Also requesting a histogram and boxplot
descr(mtcars$mpg, histogram=TRUE, boxplot=TRUE);

### To show the output as Rmd Partial in the viewer
rosetta::rosettaDescr_partial(
  rosetta::descr(
    mtcars$mpg
  )
);
```

```
### Multiple variables, including one factor
rosetta::rosettaDescr_partial(
  rosetta::descr(
    iris
  )
);
```

`descriptiveCIs` *Descriptives with confidence intervals*

Description

Descriptives with confidence intervals

Usage

```
descriptiveCIs(
  data,
  items = NULL,
  itemLabels = NULL,
  conf.level = 0.95,
  digits = 2
)

## S3 method for class 'rosettaDescriptiveCIs'
print(x, digits = attr(x, "digits"), forceKnitrOutput = FALSE, ...)
```

Arguments

`data`	The data frame holding the data, or a vector.
`items`	If supplying a data frame as `data`, the names of the columns to process.
`itemLabels`	Optionally, labels to use for the items (optionally, named, with the names corresponding to the `items`; otherwise, the order of the labels has to match the order of the items)
`conf.level`	The confidence level of the confidence intervals.
`digits`	The number of digits to round the output to.
`x`	The object to print (i.e. the object returned by `descriptiveCIs`).
`forceKnitrOutput`	Whether to force `knitr` output even when not knitting.
`...`	Any additional arguments are passed on to `knitr::kable()` or to `base::print()`.

Value

A data frame with class `rosettaDescriptiveCIs` prepended to allow printing neatly while knitting to Markdown.

Examples

```
descriptiveCIs(mtcars);
```

---

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 OmegaHat 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));
```

`factorAnalysis` *Factor analysis or principal component analysis*

Description

This is a wrapper for the psych functions `psych::pca()` and `psych::fa()` to produce output that it similar to the output produced by jamovi.

Usage

```
factorAnalysis(
  data,
  nfactors,
  items = names(data),
  rotate = "oblimin",
  covar = FALSE,
  na.rm = TRUE,
  kaiser = 1,
  loadings = TRUE,
  summary = FALSE,
  correlations = FALSE,
  modelFit = FALSE,
  eigenValues = FALSE,
  screePlot = FALSE,
  residuals = FALSE,
  itemLabels = items,
  colorLoadings = FALSE,
  fm = "minres",
  digits = 2,
  headingLevel = 3,
  ...
```

```
)

principalComponentAnalysis(
  data,
  items,
  nfactors,
  rotate = "oblimin",
  covar = FALSE,
  na.rm = TRUE,
  kaiser = 1,
  loadings = TRUE,
  summary = FALSE,
  correlations = FALSE,
  eigenValues = FALSE,
  screePlot = FALSE,
  residuals = FALSE,
  itemLabels = items,
  colorLoadings = FALSE,
  digits = 2,
  headingLevel = 3,
  ...
)

rosettaDataReduction_partial(
  x,
  digits = x$input$digits,
  headingLevel = x$input$headingLevel,
  echoPartial = FALSE,
  partialFile = NULL,
  quiet = TRUE,
  ...
)

## S3 method for class 'rosettaDataReduction'
knit_print(
  x,
  digits = x$input$digits,
  headingLevel = x$input$headingLevel,
  echoPartial = FALSE,
  partialFile = NULL,
  quiet = TRUE,
  ...
)

## S3 method for class 'rosettaDataReduction'
print(
  x,
  digits = x$input$digits,
```

```
  headingLevel = x$input$headingLevel,
  forceKnitrOutput = FALSE,
  ...
)
```

Arguments

data	The data frame that contains the items.
nfactors	The number of factors to extract, or 'eigen' to extract all factors with an eigen value higher than the number specified in kaiser. In the future, parallel can be specified here to extract the number of factors suggested by parallel analysis.
items	The items to analyse; if not specified, all variables in data will be used.
rotate	Which rotation to use; see psych::fa() for all options. The most common options are 'none' to not rotate at all; 'varimax' for an orthogonal rotation (assuming/imposing that the components or factors are not correlated); or 'oblimin' for an oblique rotation (allowing the components/factors to correlate).
covar	Whether to analyse the correlation matrix (FALSE) or the covariance matrix (TRUE).
na.rm	Whether to first remove all cases with missing values.
kaiser	The minimum eigenvalue when applying the Kaiser criterion (see nfactors).
loadings	Whether to display the component or factor loadings.
summary	Whether to display the factor or component summary.
correlations	Whether to display the correlations between factors of components.
modelFit	Whether to display the model fit Only for EFA).
eigenValues	Whether to display the eigen values.
screePlot	Whether to display the scree plot.
residuals	Whether to display the matrix with residuals.
itemLabels	Optionally, labels to use for the items (optionally, named, with the names corresponding to the items; otherwise, the order of the labels has to match the order of the items)
colorLoadings	Whether, when producing an Rmd partial (i.e. when calling the command while knitting) to colour the cells using kableExtra::kable_styling().
fm	The method to use for the factor analysis: 'fm' for Minimum Residuals; 'ml' for Maximum Likelihood; and 'pa' for Principal Factor.
digits	The number of digits to round to.
headingLevel	The number of hashes to print in front of the headings when printing while knitting
...	Any additional arguments are passed to psych::fa(), psych::pca(), to the default print method by the print method, and to rmdpartials::partial() when knitting an RMarkdown partial.
x	The object to print.
echoPartial	Whether to show the executed code in the R Markdown partial (TRUE) or not (FALSE).

`partialFile` This can be used to specify a custom partial file. The file will have object `x` available.

`quiet` Passed on to `knitr::knit()` whether it should b chatty (`FALSE`) or quiet (`TRUE`).

`forceKnitrOutput` Force knitr output.

Details

The code in these functions uses parts of the code in jamovi, written by Jonathon Love and Ravi Selker.

Value

An object with the object resulting from the call to the `psych` functions and some extracted information that will be printed.

Examples

```
### Load example dataset
data("pp15", package="rosetta");

### Get variable names with expected
### effects of a high dose of MDMA
items <-
  grep(
    "highDose_AttBeliefs_",
    names(pp15),
    value=TRUE
  );

### Do a factor analysis
rosetta::factorAnalysis(
  data = pp15,
  items = items,
  nfactors = "eigen",
  scree = TRUE
);

### To get more output, show the
### output as Rmd Partial in the viewer,
### and color/size the factor loadings
rosetta::rosettaDataReduction_partial(
  rosetta::factorAnalysis(
    data = pp15,
    items = items,
    nfactors = "eigen",
    summary = TRUE,
    correlations = TRUE,
    colorLoadings = TRUE
  )
);
```

`factorAnalysisjmv` *Factor Analysis*

Description

Factor Analysis

Usage

```
factorAnalysisjmv(
  data,
  items,
  nFactorMethod = "eigen",
  nFactors = 1,
  minEigen = 1,
  extraction = "minres",
  rotation = "oblimin",
  colorLoadings = TRUE,
  screePlot = FALSE,
  eigen = FALSE,
  factorCor = FALSE,
  factorSummary = FALSE,
  modelFit = FALSE
)
```

Arguments

`data`	the data as a data frame
`items`	a vector of strings naming the variables of interest in `data`
`nFactorMethod`	.
`nFactors`	.
`minEigen`	.
`extraction`	.
`rotation`	.
`colorLoadings`	.
`screePlot`	.
`eigen`	.
`factorCor`	.
`factorSummary`	.
`modelFit`	.

Value

A results object containing:

`results$loadings`	a html
`results$factorStats$factorSummary`	a table
`results$factorStats$factorCor`	a table
`results$modelFit$fit`	a table
`results$eigen$initEigen`	a table
`results$eigen$screePlot`	an image

`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$table`	a table

Tables can be converted to data frames with `asDF` or `as.data.frame`. For example:

```
results$table$asDF
as.data.frame(results$table)
```

`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)
```

ggBarChart *Bar chart using ggplot*

Description

This function provides a simple interface to create a ggplot2::ggplot() bar chart.

Usage

```
ggBarChart(vector, plotTheme = ggplot2::theme_bw(), ...)
```

Arguments

vector	The vector to display in the bar chart.
plotTheme	The theme to apply.
...	And additional arguments are passed to ggplot2::geom_bar().

Value

A ggplot2::ggplot() plot is returned.

Author(s)

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@behaviorchange.eu

See Also

ggplot2::geom_bar()

Examples

```
rosetta::ggBarChart(mtcars$cyl);
```

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 https://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');
```

---

histogram *Simple function to create a histogram*

Description

Simple function to create a histogram

Usage

```
histogram(vector, bins = NULL, theme = ggplot2::theme_bw())
```

Arguments

vector	A variable or vector.
bins	The number of bins; when 0, either the number of unique values in `vector` or `20`, whichever is lower.
theme	The ggplot2 theme to use.

Value

A ggplot2 plot.

Examples

```
rosetta::histogram(mtcars$mpg);
```

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,
  oddsRatios = 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(),
  headingLevel = 3
)

rosettaLogRegr_partial(
  x,
  digits = x$input$digits,
  pvalueDigits = x$input$pvalueDigits,
  headingLevel = x$input$headingLevel,
  echoPartial = FALSE,
  partialFile = NULL,
  quiet = TRUE,
  ...
)
```

```
## S3 method for class 'rosettaLogRegr'
knit_print(
  x,
  digits = x$input$digits,
  headingLevel = x$input$headingLevel,
  pvalueDigits = x$input$pvalueDigits,
  echoPartial = FALSE,
  partialFile = NULL,
  quiet = TRUE,
  ...
)

## S3 method for class 'rosettaLogRegr'
print(
  x,
  digits = x$input$digits,
  pvalueDigits = x$input$pvalueDigits,
  headingLevel = x$input$headingLevel,
  forceKnitrOutput = FALSE,
  ...
)
```

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.
oddsRatios	Whether to also present the regression coefficients as odds ratios (i.e. simply after a call to base::exp()).
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.

headingLevel
: The number of hashes to print in front of the headings

x
: The object to print (i.e. as produced by `rosetta::logRegr`).

echoPartial
: Whether to show the executed code in the R Markdown partial (TRUE) or not (FALSE).

partialFile
: This can be used to specify a custom partial file. The file will have object x available.

quiet
: Passed on to `knitr::knit()` whether it should b chatty (FALSE) or quiet (TRUE).

...
: Any additional arguments are passed to the default print method by the print method, and to `rmdpartials::partial()` when knitting an RMarkdown partial.

forceKnitrOutput
: Force knitr output.

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
rosetta::logRegr(data=mtcars, formula = vs ~ mpg);

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

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

## Not run:
### Mimic output that would be obtained
### when calling from an R Markdown file
rosetta::rosettaLogRegr_partial(
  rosetta::logRegr(
    data=mtcars,
    formula = vs ~ mpg,
    plot=TRUE
  )
);

## End(Not run)
```

`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");
```

`means` *Compute means and sums*

Description

These functions allow easily computing means and sums. Note that if you attach `rosetta` to the search path,

Usage

```
means(
  ...,
  data = NULL,
  requiredValidValues = 0,
  returnIfInvalid = NA,
  silent = FALSE
)

sums(
  ...,
  data = NULL,
  requiredValidValues = 0,
  returnIfInvalid = NA,
  silent = FALSE
)
```

Arguments

`...`	The dataframe or vectors for which to compute the means or sums. When passing a dataframe as unnamed argument (i.e. in the "dots", `...`), the means or sums for all columns in the dataframe will be computed. If you want to select one or more columns, make sure to pass the dataframe as `data`.
`data`	If a dataframe is passed as `data`, the values passed in the "dots" (`...`) will be taken as column names or indices in that dataframe. This allows easy indexing.
`requiredValidValues`	The number (if larger than 1) or proportion (if between 0 and 1) of values that have to be valid (i.e. nonmissing) before the mean or sum is returned.
`returnIfInvalid`	Which value to return for rows not meeting the criterion specified in `requiredValidValues`.
`silent`	Whether to suppress messages.

Value

The means or sums.

Examples

```
rosetta::means(mtcars$mpg, mtcars$disp, mtcars$wt);
rosetta::means(data=mtcars, 'mpg', 'disp', 'wt');
rosetta::sums(mtcars$mpg, mtcars$disp, mtcars$wt);
rosetta::sums(data=mtcars, 'mpg', 'disp', 'wt');
```

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);
```

opts *Options for the rosetta package*

Description

The rosetta::opts object contains three functions to set, get, and reset options used by the rosetta package. Use rosetta::opts$set to set options, rosetta::opts$get to get options, or rosetta::opts$reset to reset specific or all options to their default values.

Usage

```
opts
```

Format

An object of class list of length 4.

Details

It is normally not necessary to get or set rosetta options.

The following arguments can be passed:

... For rosetta::opts$set, the dots can be used to specify the options to set, in the format option = value, for example, varViewCols = c("values","level"). For rosetta::opts$reset, a list of options to be reset can be passed.

option For rosetta::opts$set, the name of the option to set.

default For rosetta::opts$get, the default value to return if the option has not been manually specified.

The following options can be set:

varViewCols The order and names of the columns to include in the variable view.

showLabellerWarning Whether to show a warning if labeller labels are encountered.

Examples

```
### Get the default columns in the variable view
rosetta::opts$get(varViewCols);

### Set it to a custom version
rosetta::opts$set(varViewCols = c("values", "level"));

### Check that it worked
rosetta::opts$get(varViewCols);

### Reset this option to its default value
rosetta::opts$reset(varViewCols);
```

```
### Check that the reset worked, too
rosetta::opts$get(varViewCols);
```

`partypanelData` *Subsets of the Party Panel 2015 dataset*

Description

This is a subsets of the Party Panel 2015 dataset. Party Panel is an annual semi-panel determinant study among Dutch nightlife patrons, where every year, the determinants of another nightlife-related risk behavior are mapped. In 2015, determinants were measured of behaviors related to using highly dosed ecstasy pills.

Usage

```
data(pp15)
```

Format

A `data.frame` with 128 columns and 829 rows. Note that many rows contain missing values; the columns and rows were taken directly from the original Party Panel dataset, and represent all participants that made it past a given behavior.

Details

The full dataset is publicly available through the Open Science Framework (https://osf.io/s4fmu/). Also see the GitLab repository (https://gitlab.com/partypanel) and the website at https://partypanel.eu.

Examples

```
data('pp15', package='rosetta');
rosetta::freq(pp15$gender);
```

`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

---

plotIMM3d *Makes 3D plots of Index of Moderated Mediation of gemm object*

Description

Makes 3D plots of Index of Moderated Mediation of gemm object

Usage

```
plotIMM3d(x, ...)
```

Arguments

x	results of gemm function
...	optional

Value

empty, directly plots all indices of mediation

---

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);
```

`prepIMM3d` *Computes Index of moderated mediation of gemm object*

Description

Computes Index of moderated mediation of gemm object

Usage

```
prepIMM3d(M1, M2, parEst = parEst, i = 1)
```

Arguments

`M1`	moderator of x-m path
`M2`	moderator of m-y path
`parEst`	parameter estimates from lavaan results
`i`	index of vector of mediators names

Value

vector of index of moderated mediation with CI limits for a given mediator

`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

recode *Recode a Variable (car version)*

Description

This function is from the **car** package. Please see that help page for details: `car::recode()`.

Usage

```
recode(var, recodes, as.factor, as.numeric = TRUE, levels)
```

Arguments

var	numeric vector, character vector, or factor.
recodes	character string of recode specifications: see below.
as.factor	return a factor; default is TRUE if var is a factor, FALSE otherwise.
as.numeric	if TRUE (the default), and as.factor is FALSE, then the result will be coerced to numeric if all values in the result are numerals—i.e., represent numbers.
levels	an optional argument specifying the order of the levels in the returned factor; the default is to use the sort order of the level names.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Examples

```
x<-rep(1:3,3)
x
rosetta::recode(
  x,
  "c(1,2)='A'; else='B'"
);
rosetta::recode(
  x,
  "1:2='A'; 3='B'"
);
```

`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; for more functionality, use the base R function `lm` or e.g. the `lme4` package.

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,
  headingLevel = 3,
  env = parent.frame()
)

rosettaRegr_partial(
```

```
  x,
  digits = x$input$digits,
  pvalueDigits = x$input$pvalueDigits,
  headingLevel = x$input$headingLevel,
  echoPartial = FALSE,
  partialFile = NULL,
  quiet = TRUE,
  ...
)

## S3 method for class 'rosettaRegr'
knit_print(
  x,
  digits = x$input$digits,
  headingLevel = x$input$headingLevel,
  pvalueDigits = x$input$pvalueDigits,
  echoPartial = FALSE,
  partialFile = NULL,
  quiet = TRUE,
  ...
)

## S3 method for class 'rosettaRegr'
print(
  x,
  digits = x$input$digits,
  pvalueDigits = x$input$pvalueDigits,
  headingLevel = x$input$headingLevel,
  forceKnitrOutput = FALSE,
  ...
)

## S3 method for class 'rosettaRegr'
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. They are not printed.
`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.
`headingLevel`	The number of hashes to print in front of the headings when printing while knitting
`env`	The enviroment where to evaluate the formula.
`x`	The object to print (i.e. as produced by `regr`).
`echoPartial`	Whether to show the executed code in the R Markdown partial (`TRUE`) or not (`FALSE`).
`partialFile`	This can be used to specify a custom partial file. The file will have object `x` available.
`quiet`	Passed on to `knitr::knit()` whether it should b chatty (`FALSE`) or quiet (`TRUE`).
`...`	Any additional arguments are passed to the default print method by the print method, and to `rmdpartials::partial()` when knitting an RMarkdown partial.
`forceKnitrOutput`	Force knitr output.

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
rosetta::regr(age ~ circumference, dat=Orange);

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

## Not run:
### An example with an interaction term, showing in the
### viewer
rosetta::rosettaRegr_partial(
  rosetta::regr(
    mpg ~ wt + hp + wt:hp,
    dat=mtcars,
    coefficients = "raw",
    plot=TRUE,
    collinearity=TRUE
  )
);

## End(Not run)
```

`reliability` *Conduct reliability analyses with output similar to jamovi and SPSS*

Description

The `reliability()` analysis is the only one most users will need. It tries to apply best practices by, as much as possible, complementing point estimates with confidence intervals.

Usage

```
reliability(
  data,
  items = NULL,
  scaleStructure = TRUE,
  descriptives = FALSE,
  itemLevel = FALSE,
  scatterMatrix = FALSE,
  scatterMatrixArgs = list(progress = FALSE),
  digits = 2,
  conf.level = 0.95,
  itemLabels = NULL,
  itemOmittedCorsWithRest = FALSE,
  itemOmittedCorsWithTotal = FALSE,
  alphaOmittedCIs = FALSE,
```

```
  omegaFromMBESS = FALSE,
  omegaFromPsych = TRUE,
  ordinal = FALSE,
  headingLevel = 3,
  ...
)

rosettaReliability_partial(
  x,
  digits = x$digits,
  headingLevel = x$headingLevel,
  printPlots = TRUE,
  echoPartial = FALSE,
  partialFile = NULL,
  quiet = TRUE,
  ...
)

## S3 method for class 'rosettaReliability'
knit_print(
  x,
  digits = x$digits,
  headingLevel = x$headingLevel,
  printPlots = TRUE,
  echoPartial = FALSE,
  partialFile = NULL,
  quiet = TRUE,
  ...
)

## S3 method for class 'rosettaReliability'
print(
  x,
  digits = x$digits,
  headingLevel = x$headingLevel,
  forceKnitrOutput = FALSE,
  printPlots = TRUE,
  ...
)
```

Arguments

`data`	The data frame
`items`	The items (if omitted, all columns are used)
`scaleStructure`	Whether to include scale-level estimates using `ufs::scaleStructure()`
`descriptives`	Whether to include mean and standard deviation eastimates and their confidence intervals
`itemLevel`	Whether to include item-level internal consistency estimates

scatterMatrix, scatterMatrixArgs
: Whether to produce a scatter matrix, and the arguments to pass to the scatterMatrix() function.

digits
: The number of digits to round the result to

conf.level
: The confidence level of confidence intervals

itemLabels
: Optionally, labels to use for the items (optionally, named, with the names corresponding to the items; otherwise, the order of the labels has to match the order of the items)

itemOmittedCorsWithRest, itemOmittedCorsWithTotal
: Whether to include each item's correlations with, respectively, the scale with that item omitted, or the full scale.

alphaOmittedCIs
: Whether to include the confidence intervals for the Coefficient Alpha estimates with the item omitted.

omegaFromMBESS, omegaFromPsych
: Whether to include omega from MBESS and/or psych

ordinal
: Wheher to set poly=TRUE when calling ufs::scaleStructure(), which will compute the polychoric correlation matrix to provide the scale estimates assuming ordinal-level items. Note that this may throw a variety of errors from within the psych package if the data are somehow not what psych expects

headingLevel
: The number of hashes to print in front of the headings when printing while knitting

...
: Any additional arguments are passed to ufs::scaleStructure() by reliability, to the default print method by print.reliability, and to rmdpartials::partial() when knitting an RMarkdown partial.

x
: The object to print

printPlots
: Whether to print plots (can be used to suppress plots, which can be useful sometimes)

echoPartial
: Whether to show the executed code in the R Markdown partial (TRUE) or not (FALSE).

partialFile
: This can be used to specify a custom partial file. The file will have object x available.

quiet
: Passed on to knitr::knit() whether it should b chatty (FALSE) or quiet (TRUE).

forceKnitrOutput
: Force knitr output

Details

The rosettaReliability object that is returned has its own print() method, that, when using knitr, will use the rmdpartials package to insert an RMarkdown partial. That partial is created using rosettaReliability_partial(), which is also called by a specific knit_print() method.

Value

An object with all results

Examples

```
### These examples aren't run during tests
### because they can take quite long
## Not run:
### Simple example with only main reliability results
data(pp15, package="rosetta");
rosetta::reliability(
  pp15,
  c(
    "highDose_AttGeneral_good",
    "highDose_AttGeneral_prettig",
    "highDose_AttGeneral_slim",
    "highDose_AttGeneral_gezond",
    "highDose_AttGeneral_spannend"
  )
);

### More extensive example with an RMarkdown partial that
### displays in the viewer
rosetta::rosettaReliability_partial(
  rosetta::reliability(
    attitude,
    descriptives = TRUE,
    itemLevel = TRUE,
    scatterMatrix = TRUE
  )
);

## End(Not run)
```

---

varView *Variable View*

Description

This function provides an overview of the variables in a dataframe, allowing efficient inspection of the factor levels, ranges for numeric variables, and numbers of missing values.

Usage

```
varView(
  data,
  columns = names(data),
  varViewCols = rosetta::opts$get(varViewCols),
  varViewRownames = TRUE,
  maxLevels = 10,
  truncLevelsAt = 50,
  showLabellerWarning = rosetta::opts$get(showLabellerWarning),
```

```
  output = rosetta::opts$get("tableOutput")
)

## S3 method for class 'rosettaVarView'
print(x, output = attr(x, "output"), ...)
```

Arguments

`data`	The dataframe containing the variables to view.
`columns`	The columns to include.
`varViewCols`	The columns of the variable view.
`varViewRownames`	Whether to set the variable names as row names of the variable view dataframe that is returned.
`maxLevels`	For factors, the maximum number of levels to show.
`truncLevelsAt`	For factors levels, the number of characters at which to truncate.
`showLabellerWarning`	Whether to show a warning if labeller labels are encountered.
`output`	A character vector containing one or more of "`console`", "`viewer`", and one or more filenames in existing directories. If `output` contains `viewer` and RStudio is used, the variable view is shown in the RStudio viewer.
`x`	The varView data frame to print.
`...`	Any additional arguments are passed along to the `print.data.frame()` function.

Value

A dataframe with the variable view.

Author(s)

Gjalt-Jorn Peters & Melissa Gordon Wolf

Examples

```
### The default variable view
rosetta::varView(iris);

### Only for a few variables in the dataset
rosetta::varView(iris, columns=c("Sepal.Length", "Species"));

### Set some variable and value labels using the `labelled`
### standard, which is also used by `haven`
dat <- iris;
attr(dat$Sepal.Length, "label") <- "Sepal length";
attr(dat$Sepal.Length, "labels") <-
  c('one' = 1,
    'two' = 2,
```

```
    'three' = 3);

### varView automatically recognizes and shows these, adding
### a 'label' column
rosetta::varView(dat);

### You can also specify that you only want to see some columns
### in the variable view
rosetta::varView(dat,
                 varViewCols = c('label', 'values', 'level'));
```

Index