Package 'JWileymisc'

May 7, 2026

Title Miscellaneous Utilities and Functions

Version 1.4.4

URL https://joshuawiley.com/JWileymisc/,
https://github.com/JWiley/JWileymisc

BugReports https://github.com/JWiley/JWileymisc/issues

Description Miscellaneous tools and functions,
including: generate descriptive statistics tables,
format output, visualize relations among variables or check
distributions, and generic functions for residual and
model diagnostics.

License GPL (>= 3)

Depends R (>= 4.1.0)

Imports stats, utils, MASS, multcompView, emmeans, data.table (>=
1.14.8), graphics, grid, ggplot2 (>= 3.4.3), scales, ggpubr,
mgcv, mice, methods, psych, robustbase, gamlss, lavaan (>=
0.6-16), VGAM (>= 1.1-9), lme4, extraoperators (>= 0.1.1),
digest, fst, rlang

Suggests foreach, testthat (>= 3.1.10), covr, withr, knitr, rmarkdown,
pander, GPArotation, rms

Encoding UTF-8

LazyData true

Config/testthat/edition 3

VignetteBuilder knitr

RoxygenNote 7.3.2

NeedsCompilation no

Author Joshua F. Wiley [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0271-6702>)

Maintainer Joshua F. Wiley <jwiley.psych@gmail.com>

Repository CRAN

Date/Publication 2025-09-06 05:12:39 UTC

Contents

`.fround` *Function to round and format a number*

Description

Function to round and format a number

Usage

```
.fround(x, digits)
```

Arguments

`x`	the data to round and format
`digits`	the number of digits to used

Value

a character vector

.quantilePercentiles *Internal Function to Calculate Quantiles*

Description

Function calculates smoothing spline quantiles or linear quantiles as a fall back. Not intended for general use. Expected predicted and residual data. Exported to support related packages.

Usage

```
.quantilePercentiles(
  data,
  Mid = 0.5,
  LL = 0.1,
  UL = 0.9,
  na.rm = TRUE,
  cut = 8L
)
```

Arguments

data	A dataset of predicted and residual values. Assumed from some sort of (probably parametric) model.
Mid	The middle limit for prediction. Defaults to .5 to give the median.
LL	The lower limit for prediction. Defaults to .1 to give the 10th percentile.
UL	The upper limit for prediction. Defaults to .9 to give the 90th percentile.
na.rm	A logical whether to remove missing values. Defaults to TRUE
cut	An integer, how many unique predicted values there have to be at least for it to use quantile regression or treat the predicted values as discrete. Defaults to 8.

Value

A data.table with the scores and predicted LL and UL, possibly missing if quantile regression models do not converge.

aces_daily *Multilevel Daily Data Example*

Description

A data frame drawn from a daily diary study, conducted at Monash University in 2017 where young adults old completed measures up to three times per day (morning, afternoon, and evening) for about 12 days. Thus each participant contributed about 36 observations to the dataset. To protect participant confidentiality and anonymity, the data used here were simulated from the original data, but in such a way as to preserve the relations among variables and most features of the raw data.

Usage

```
aces_daily
```

Format

A data frame containing 19 variables.

UserID A unique identifier for each individual

SurveyDay The date each observation occured on

SurveyInteger The survey coded as an integer (1 = morning, 2 = afternoon, 3 = evening)

SurveyStartTimec11 Survey start time, centered at time since 11am

Female A 0 or 1 variable, where 1 = female and 0 = male

Age Participant age in years, top coded at 25

BornAUS A 0 or 1 variable where 1 = born in Australia and 0 = born outside of Australia

SES_1 Participants subjective SES, bottom coded at 4 and top coded at 8

EDU Participants level of education (1 = university graduate or higher, 0 = less than university graduate

SOLs Self-reported sleep onset latency in minutes, morning survey only

WASONs Self-reported number of wakenings after sleep onset, top coded at 4, morning survey only

STRESS Overall stress ratings on a 0–10 scale, repeated 3x daily

SUPPORT Overall social support ratings on a 0–10 scale, repeated 3x daily

PosAff Positive affect ratings on a 1–5 scale, repeated 3x daily

NegAff Negative affect ratings on a 1–5 scale, repeated 3x daily

COPEPrb Problem focused coping on a 1–4 scale, repeated 1x daily at the evening survey

COPEPrc Emotional processing coping on a 1–4 scale, repeated 1x daily at the evening survey

COPEExp Emotional expression coping on a 1–4 scale, repeated 1x daily at the evening survey

COPEDis Mental disengagement coping on a 1–4 scale, repeated 1x daily at the evening survey

---

APAStyler *A generic function for pretty printing in (semi) APA Style*

Description

A generic function for pretty printing in (semi) APA Style

Usage

```
APAStyler(object, ...)
```

Arguments

`object`	An object with a class matching one of the methods
`...`	Additional argiuments passed on to methods.

`APAStyler.list` *APAStyler method for lists*

Description

This assumes that all the objects in a list have the same class and that an `APAStyler` method exists for that class.

Usage

```
## S3 method for class 'list'
APAStyler(object, ...)
```

Arguments

`object`	A list in this case, where each element is another known class.
`...`	Additional arguments.

Value

Styled results.

Examples

```
## Not run:
m1 <- lm(mpg ~ qsec * hp, data = mtcars)
m2 <- lm(mpg ~ qsec + hp, data = mtcars)
m3 <- lm(mpg ~ am + vs, data = mtcars)
mt1 <- modelTest(m1)
mt2 <- modelTest(m2)
mt3 <- modelTest(m3)

## styling regression models
APAStyler(list(m1, m2))

## modelTest objects get merged
APAStyler(list(mt1, mt2))

## the models can be named by passing a named list
## including "special" characters using backticks, like spaces
APAStyler(list(Full = mt1, Reduced = mt2))
APAStyler(list(Full = mt1, Reduced = mt2, `Alternate Model` = mt3))

## you can customize the way output is presented
APAStyler(list(mt1, mt2), format = list(
  FixedEffects = "%s, %s\n(%s, %s)",
  EffectSizes = "Cohen's f2 = %s (%s)"))

## clean up
```

```
rm(m1, m2, m3, mt1, mt2, mt3)

## End(Not run)
```

`APAStyler.lm` *APAStyler method for linear models*

Description

APAStyler method for linear models

Usage

```
## S3 method for class 'lm'
APAStyler(object, digits = 2, pdigits, file, print = TRUE, ...)
```

Arguments

`object`	A `lm` object
`digits`	The number of digits to round results to. Defaults to 2.
`pdigits`	The number of digits to use for p values. Defaults to digits + 1 if missing.
`file`	An optional argument indicating whether the output should be written to a file.
`print`	A logical argument, whether or not to print results to screen. This is distinct from saving them to a file. Defaults to `TRUE` for back compatibility.
`...`	Additional argiuments passed on to `write.table`.

`APAStyler.mira` *A generic function for pretty printing in (semi) APA Style*

Description

A generic function for pretty printing in (semi) APA Style

Usage

```
## S3 method for class 'mira'
APAStyler(object, lmobject, digits = 2, pdigits, print = TRUE, file, ...)
```

Arguments

object	mira object
lmobject	an lm object the degrees of freedom of which can be used for conservative F tests
digits	The number of digits to round results to. Defaults to 2.
pdigits	The number of digits to use for p values. Defaults to digits + 1 if missing.
print	A logical argument, whether or not to print results to screen. This is distinct from saving them to a file. Defaults to TRUE for back compatibility.
file	An optional argument indicating whether the output should be written to a file.
...	Additional argiuments passed on to write.table.

APAStyler.modelTest.lm

APAStyler method for model tests from a linear model

Description

APAStyler method for model tests from a linear model

Usage

```
## S3 method for class 'modelTest.lm'
APAStyler(
  object,
  format = list(FixedEffects = c("%s%s [%s, %s]"), EffectSizes = c("f2 = %s, %s")),
  digits = 2,
  pcontrol = list(digits = 3, stars = TRUE, includeP = FALSE, includeSign = FALSE,
    dropLeadingZero = TRUE),
  ...
)
```

Arguments

object	A modelTest.lm class object, results from running modelTest() function on a class lm object.
format	A list giving the formatting style to be used for the fixed effecvts and effect sizes.
digits	A numeric value indicating the number of digits to print. This is still in early implementation stages and currently does not change all parts of the output (which default to 2 decimals per APA style).
pcontrol	A list controlling how p values are formatted.
...	Additional arguments.

Value

Styled results.

Examples

```
m1 <- lm(mpg ~ qsec * hp, data = mtcars)
APAStyler(modelTest(m1))

APAStyler(modelTest(m1),
format = list(
  FixedEffects = "%s, %s\n(%s, %s)",
  EffectSizes = "Cohen's f2 = %s (%s)"),
pcontrol = list(digits = 4,
  stars = FALSE, includeP = TRUE,
  includeSign = TRUE,
  dropLeadingZero = TRUE))

## clean up
rm(m1)
```

APAStyler.modelTest.vglm

APAStyler method for model tests from a vglm multinomial model

---

Description

APAStyler method for model tests from a vglm multinomial model

Usage

```
## S3 method for class 'modelTest.vglm'
APAStyler(
  object,
  format = list(FixedEffects = c("%s%s [%s, %s]"), EffectSizes =
    c("Chi-square (df=%s) = %s, %s")),
  digits = 2,
  pcontrol = list(digits = 3, stars = TRUE, includeP = FALSE, includeSign = FALSE,
    dropLeadingZero = TRUE),
  OR = TRUE,
  ...
)
```

Arguments

object	A `modelTest.vglm` class object, results from running `modelTest()` function on a class `vglm` object with a multinomial family
format	A list giving the formatting style to be used for the fixed effects and effect sizes.
digits	A numeric value indicating the number of digits to print. This is still in early implementation stages and currently does not change all parts of the output (which default to 2 decimals per APA style).
pcontrol	A list controlling how p values are formatted.

OR	a logical value whether to report odds ratios and 95 percent confidence intervals, if TRUE, or regression coefficients on the logit scale with standard errors, if FALSE.
...	Additional arguments.

Value

Styled results.

Examples

```
mtcars$cyl <- factor(mtcars$cyl)
m <- VGAM::vglm(cyl ~ qsec,
  family = VGAM::multinomial(), data = mtcars)
mt <- modelTest(m)

APAStyler(mt)

APAStyler(mt, OR = FALSE)

## clean up
rm(m, mt, mtcars)

## Not run:
mtcars$cyl <- factor(mtcars$cyl)
mtcars$am <- factor(mtcars$am)
m <- VGAM::vglm(cyl ~ qsec,
  family = VGAM::multinomial(), data = mtcars)
APAStyler(modelTest(m))

m <- VGAM::vglm(cyl ~ scale(qsec),
  family = VGAM::multinomial(), data = mtcars)
APAStyler(modelTest(m))

m2 <- VGAM::vglm(cyl ~ factor(vs) * scale(qsec),
  family = VGAM::multinomial(), data = mtcars)
APAStyler(modelTest(m2))

m <- VGAM::vglm(Species ~ Sepal.Length,
  family = VGAM::multinomial(), data = iris)
APAStyler(modelTest(m))

set.seed(1234)
sampdata <- data.frame(
  Outcome = factor(sample(letters[1:3], 20 * 9, TRUE)),
  C1 = rnorm(20 * 9),
  D3 = sample(paste0("L", 1:3), 20 * 9, TRUE))

m <- VGAM::vglm(Outcome ~ factor(D3),
  family = VGAM::multinomial(), data = sampdata)
APAStyler(modelTest(m))
```

```
m <- VGAM::vglm(Outcome ~ factor(D3) + C1,
  family = VGAM::multinomial(), data = sampdata)
APAStyler(modelTest(m))

## End(Not run)
```

APAStyler.SEMSummary *A generic function for pretty printing in (semi) APA Style*

Description

A generic function for pretty printing in (semi) APA Style

Usage

```
## S3 method for class 'SEMSummary'
APAStyler(
  object,
  digits = 2,
  type = c("cov", "cor", "both"),
  stars = FALSE,
  file = ifelse(.Platform$OS.type == "windows", "clipboard", FALSE),
  sep = "\t",
  print = TRUE,
  ...
)
```

Arguments

object	SEMSummary object
digits	The number of digits to round results to. Defaults to 2.
type	A character vector giving what to print. Defaults to 'cov', the covariances. Other options are 'cor' and 'both'.
stars	A logical value whether to include significance values as stars (*** p < .001, ** p < .01, * p < .05).
file	An optional argument indicating whether the output should be written to a file.
sep	Character what the separator for the table should be. Defaults to tabs.
print	A logical argument, whether or not to print results to screen. This is distinct from saving them to a file. Defaults to TRUE for back compatibility.
...	Additional argiuments passed on to write.table.

Examples

```
m <- SEMSummary(~., data = mtcars)
APAStyler(m, type = "cor", stars = FALSE, file = FALSE)
APAStyler(m, type = "cov", stars = FALSE, file = FALSE)
APAStyler(m, type = "both", stars = FALSE, file = FALSE)
APAStyler(m, type = "cor", stars = TRUE, file = FALSE)
APAStyler(m, type = "cov", stars = TRUE, file = FALSE)
APAStyler(m, type = "both", stars = TRUE, file = FALSE)
```

---

as.na *Coerces vectors to missing*

Description

Given a vector, convert it to missing (NA) values, where the class of the missing matches the input class. Currently supports character, logical, integer, factor, numeric, times (from **chron**), Date, POSIXct, POSIXlt, and zoo (from **zoo**), and haven labelled from **haven**.

Usage

```
as.na(x)
```

Arguments

x	A vector to convert to missing (NA)

Value

a vector the same length as the input with missing values of the same class

Examples

```
str(as.na(1L:5L))
str(as.na(rnorm(5)))
str(as.na(c(TRUE, FALSE)))
str(as.na(as.Date("2017-01-01")))
```

cd *Change directory*

Description

The function takes a path and changes the current working directory to the path. If the directory specified in the path does not currently exist, it will be created.

Usage

```
cd(base, pre, num)
```

Arguments

base	a character string with the base path to the directory. This is required.
pre	an optional character string with the prefix to add to the base path. Non character strings will be coerced to character class.
num	an optional character string, prefixed by pre. Non character strings will be coerced to character class.

Details

The function has been designed to be platform independent, although it has had limited testing. Path creation is done using file.path, the existence of the directory is checked using file.exists and the directory created with dir.create. Only the first argument, is required. The other optional arguments are handy when one wants to create many similar directories with a common base.

Value

NULL, changes the current working directory

Examples

```
## Not run:
# an example just using the base
cd("~/testdir")

# an example using the optional arguments
base <- "~/testdir"
pre <- "test_"

cd(base, pre, 1)
cd(base, pre, 2)

## End(Not run)
```

`compareIVs` *Compares the effects of various independent variables on dependent variables*

Description

Utility to estimate the unadjusted, covariate adjusted, and multivariate adjusted unique contributions of one or more IVs on one or more DVs

Usage

```
compareIVs(
  dv,
  type,
  iv,
  covariates = character(),
  data,
  multivariate = FALSE,
  ...
)
```

Arguments

`dv`	A character string or vector of the depentent variable(s)
`type`	A character string or vector indicating the type of dependent variable(s)
`iv`	A character string or vector giving the IV(s)
`covariates`	A character string or vector giving the covariate(s)
`data`	The data to be used for analysis
`multivariate`	A logical value whether to have models with all IVs simultaneously.
`...`	Additional arguments passed on to the internal function, `.runIt`.

Value

A list with all the model results.

Examples

```
test1 <- compareIVs(
  dv = c("mpg", "disp"),
  type = c("normal", "normal"),
  iv = c("hp", "qsec"),
  covariates = "am",
  data = mtcars, multivariate = TRUE)
test1$OverallSummary
rm(test1)
```

compressed RDS *Save and read RDS functions for using multithreaded "ZSTD" or "LZ4" compression*

Description

Save and read RDS functions for using multithreaded "ZSTD" or "LZ4" compression

Usage

```
saveRDSfst(object, filename = "", compression = 100, algorithm = "ZSTD")

readRDSfst(filename)
```

Arguments

object	An R object to be saved.
filename	A character string giving the filename of the object on disk (to save to or read from)
compression	A numeric value between 0 and 100 indicating the amount of compression. Defaults to 100, the highest level of compression. 0 gives the lowest compression.
algorithm	A character string of the type of compression to use. Defaults to "ZSTD" which is better compression but slower. The only other option is "LZ4" which is faster but may provide less compression.

Details

By default, `saveRDS()` does not have multithreaded compression built in. These functions use "ZSTD" or "LZ4" compression via the `fst` package for multithreaded compression and decompression with good performance. To save them, objects are serialized, compressed, and then saved using `saveRDS()`. To read them, objects are read in using `readRDS()`, decompressed, and then unserialized. Hashing is used to verify the results. `saveRDS()` is performed using `version = 3`, so it will not work on older versions of `R`.

Value

`saveRDSfst()` is called for its side effect of saving a file to disk. The original `R` object if using `readRDSfst()`.

Examples

```
saveRDSfst(mtcars, filename = file.path(tempdir(), "mtcars.RDS"))

saveRDSfst(mtcars, filename = file.path(tempdir(), "mtcars.RDS"))
readRDSfst(file.path(tempdir(), "mtcars.RDS"))
```

cor2cov — *Convert a correlation matrix and standard deviations to a covariance matrix*

Description

This is a simple function designed to convert a correlation matrix (standardized covariance matrix) back to a covariance matrix. It is the opposite of `cov2cor`.

Usage

```
cor2cov(V, sigma)
```

Arguments

`V`	an n x n correlation matrix. Should be numeric, square, and symmetric.
`sigma`	an n length vector of the standard deviations. The length of the vector must match the number of columns in the correlation matrix.

Value

an n x n covariance matrix

See Also

`cov2cor`

Examples

```
# using a built in dataset
cor2cov(cor(longley), sapply(longley, sd))

# should match the above covariance matarix
cov(longley)
all.equal(cov(longley), cor2cov(cor(longley), sapply(longley, sd)))
```

corOK — *Return a non-missing correlation matrix*

Description

Given a square, symmetric matrix (such as a correlation matrix) this function tries to drop the fewest possible number of variables to return a (square, symmetric) matrix with no missing cells.

Usage

```
corOK(x, maxiter = 100)
```

Arguments

x	a square, symmetric matrix or object coercable to such (such as a data frame).
maxiter	a number indicating the maximum number of iterations, currently as a sanity check. See details.

Details

The assumption that x is square and symmetric comes because it is assumed that the number of missing cells for a given column are identical to that of the corresponding row. corOK finds the column with the most missing values, and drops that (and its corresponding row), and continues on in like manner until the matrix has no missing values. Although this was intended for a correlation matrix, it could be used on other types of matrices. Note that because corOK uses an iterative method, it can be slow when many columns/rows need to be removed. For the intended use (correlation matrices) there probably should not be many missing. As a sanity check and to prevent tediously long computations, the maximum number of iterations can be set.

Value

A list with two elements

x	The complete non missing matrix.
keep.indices	A vector of the columns and rows from the original matrix to be kept (i.e., that are nonmissing).

Examples

```
cormat <- cor(iris[, -5])
# set missing
cormat[cbind(c(1,2), c(2,1))] <- NA

# print
cormat

# return complete
corOK(cormat)

# using maximum iterations
corOK(cormat, maxiter=0)

# clean up
rm(cormat)
```

---

corplot *Heatmap of a Correlation Matrix*

Description

This function creates a heatmap of a correlation matrix using **ggplot2**.

Usage

```
corplot(
  x,
  coverage,
  pvalues,
  type = c("both", "cor", "p", "coverage"),
  digits = 2,
  order = c("cluster", "asis"),
  ...,
  control.grobs = list()
)
```

Arguments

x	A correlation matrix or some other square symmetric matrix.
coverage	An (optional) matrix with the same dimensions as x giving the proportion of data present. Particularly useful when the correlation matrix is a pairwise present.
pvalues	An (optional) matrix with the same dimensions as x giving the p values for each correlation. To show, use plot = "p".
type	A character string indicating what to show on top of the heatmap. Can be 'coverage', in which case bubble points show coverage; 'p', in which case p values are shown; 'cor', in which case correlations are shown; or 'both', in which case both correlations and p-values are shown. Only has an effect if a coverage (or pvalue) matrix is passed also. Defaults to cor.
digits	The number of digits to round to when printing the correlations on the heatmap. Text is suppressed when a coverage matrix is passed and points = TRUE.
order	A character string indicating how to order the resulting plot. Defaults to 'cluster' which uses hierarchical clustering to sensibly order the variables. The other option is 'asis' in which case the matrix is plotted in the order it is passed.
...	Additional arguments currently only passed to hclust and corOK.
control.grobs	A list of additional quote()d options to customize the ggplot2 output.

Details

The actual plot is created using ggplot2 and geom_tile. In addition to creating the plot, the variables are ordered based on a hierarchical clustering of the correlation matrix. Specifically, 1 - x is used as the distance matrix. If coverage is passed, will also add a bubble plot with the area proportional to the proportion of data present for any given cell. Defaults for ggplot2 are set, but it is possible to use a named list of quote()d ggplot calls to override all defaults. This is not expected for typical use. Particularly main, points, and text as these rely on internal variable names; however, labels, the gradient color, and area scaling can be adjusted more safely.

Value

Primarily called for the side effect of creating a plot. However, the ggplot2 plot object is returned, so it can be saved, replotted, edited, etc.

Examples

```
# example plotting the correlation matrix from the
# mtcars dataset
corplot(cor(mtcars))

dat <- as.matrix(iris[, 1:4])

# randomly set 25% of the data to missing
set.seed(10)
dat[sample(length(dat), length(dat) * .25)] <- NA
cor(dat, use = "pair")
cor(dat, use = "complete")

# create a summary of the data (including coverage matrix)
sdat <- SEMSummary(~ ., data = dat, use = "pair")
str(sdat)
# using the plot method for SEMSummary (which basically just calls corplot)
## getting correlations above diagonal and p values below diagonal#'
plot(sdat)

## get correlations only
plot(sdat, type = "cor")

## showing coverage
plot(sdat, type = "coverage")

# use the control.grobs argument to adjust the coverage scaling
# to go from 0 to 1 rather than the range of coverage
corplot(x = sdat$sSigma, coverage = sdat$coverage,
  type = "coverage",
  control.grobs = list(area = quote(scale_size_area(limits = c(0, 1))))
)

# also works with plot() on a SEMSummary
plot(x = sdat, type = "coverage",
  control.grobs = list(area = quote(scale_size_area(limits = c(0, 1))))
)

rm(dat, sdat)
```

`cramerV` *Calculate Phi or Cramer's V effect size*

Description

Simple function to calculate effect sizes for frequency tables.

Usage

```
cramerV(x)
```

Arguments

x	A frequency table, such as from xtabs().

Value

A numeric value with Phi for 2 x 2 tables or Cramer's V for tables larger than 2 x 2.

Examples

```
cramerV(xtabs(~ am + vs, data = mtcars))
cramerV(xtabs(~ cyl + vs, data = mtcars))
cramerV(xtabs(~ cyl + am, data = mtcars))
```

---

density_inversion *KDE-based Inverse CDF Sampling for Synthetic Data*

Description

Generate synthetic samples that try to follow an observed distribution by inverting a smooth kernel density estimate (KDE).

Usage

```
density_inversion(x, n, KDEn = 100, seed = NULL)
```

Arguments

x	Numeric vector containing the observed data. Should be numeric or integer. Vector only (no datasets at this stage).
n	Integer scalar, the number of synthetic samples to generate.
KDEn	Integer scalar passed to [stats::density()] as the 'n' grid size for the KDE; larger values yield a finer grid for integration (default '100').
seed	Optional integer to set the random seed for reproducibility.

Details

This utility estimates a univariate density for the input vector, numerically integrates it to obtain a smooth cumulative distribution function (CDF), and then samples synthetic values by inverting that CDF for uniformly distributed probabilities. The result is a set of values that follow the shape of the observed data without reproducing exact observations.

The KDE is fit using [stats::density()] with bandwidth rule 'bw = "nrd0"'. The CDF is computed via trapezoidal integration of the KDE on its grid and normalized to \[0, 1\]. An inverse-CDF function is obtained via linear interpolation ([stats::approxfun()]), which is then evaluated at 'n' independent uniforms from \[0, 1\].

This approach produces smooth synthetic samples that closely match the observed density.

Value

A numeric vector of length ‘n‘ containing the synthetic values.

See Also

[stats::density()], [stats::approxfun()], [stats::runif()].

Examples

```
sim_iris <- density_inversion(iris$Sepal.Length, n = 200, seed = 1234)

hist(iris$Sepal.Length)
hist(sim_iris)

# cleanup
rm(sim_iris)
```

---

diffCircular *Calculate the Circular Difference*

Description

Calculate the Circular Difference

Usage

```
diffCircular(x, y, max)
```

Arguments

x	Numeric or integer values
y	Numeric or integer values
max	the theoretical maximum (e.g., if degrees, 360; if hours, 24; etc.).

Value

A value with the circular difference. This will always be positive if defined.

Examples

```
diffCircular(330, 30, max = 360)
diffCircular(22, 1, max = 24)
diffCircular(c(22, 23, 21, 22), c(1, 1, 23, 14), max = 24)
```

egltable *Function makes nice tables*

Description

Give a dataset and a list of variables, or just the data in the vars. For best results, convert categorical variables into factors. Provides a table of estimated descriptive statistics optionally by group levels.

Usage

```
egltable(
  vars,
  g,
  data,
  idvar,
  strict = TRUE,
  parametric = TRUE,
  paired = FALSE,
  simChisq = FALSE,
  sims = 1000000L
)
```

Arguments

vars	Either an index (numeric or character) of variables to access from the data argument, or the data to be described itself.
g	A variable used tou group/separate the data prior to calculating descriptive statistics.
data	optional argument of the dataset containing the variables to be described.
idvar	A character string indicating the variable name of the ID variable. Not currently used, but will eventually support egltable supporting repeated measures data.
strict	Logical, whether to strictly follow the type of each variable, or to assume categorical if the number of unique values is less than or equal to 3.
parametric	Logical whether to use parametric tests in the case of multiple groups to test for differences. Only applies to continuous variables. If TRUE, the default, uses one-way ANOVA, and a F test. If FALSE, uses the Kruskal-Wallis test.
paired	Logical whether the data are paired or not. Defaults to FALSE. If TRUE, the grouping variable, g, must have two levels and idvar must be specified. When used a paired t-test is used for parametric, continuous data and a Wilcoxon test for paired non parametric, continuous data and a McNemar chi square test is used for categorical data.
simChisq	Logical whether to estimate p-values for chi-square test for categorical data when there are multiple groups, by simulation. Defaults to FALSE. Useful when there are small cells as will provide a more accurate test in extreme cases, similar to Fisher Exact Test but generalizing to large dimension of tables.

`sims` Integer for the number of simulations to be used to estimate p-values for the chi-square tests for categorical variables when there are multiple groups. Defaults to one million (`1e6L`).

Value

A data frame of the table.

Examples

```
egltable(iris)
egltable(colnames(iris)[1:4], "Species", data = iris)
egltable(iris, parametric = FALSE)
egltable(colnames(iris)[1:4], "Species", iris,
  parametric = FALSE)
egltable(colnames(iris)[1:4], "Species", iris,
  parametric = c(TRUE, TRUE, FALSE, FALSE))
egltable(colnames(iris)[1:4], "Species", iris,
  parametric = c(TRUE, TRUE, FALSE, FALSE), simChisq=TRUE)

diris <- data.table::as.data.table(iris)
egltable("Sepal.Length", g = "Species", data = diris)

tmp <- mtcars
tmp$cyl <- factor(tmp$cyl)
tmp$am <- factor(tmp$am, levels = 0:1)

egltable(c("mpg", "hp"), "vs", tmp)
egltable(c("mpg", "hp"), "am", tmp)
egltable(c("am", "cyl"), "vs", tmp)

tests <- with(sleep,
    wilcox.test(extra[group == 1],
           extra[group == 2], paired = TRUE))
str(tests)

## example with paired data
egltable(c("extra"), g = "group", data = sleep, idvar = "ID", paired = TRUE)

## what happens when ignoring pairing (p-value off)
# egltable(c("extra"), g = "group", data = sleep, idvar = "ID")

## paired categorical data example
## using data on chick weights to create categorical data
tmp <- subset(ChickWeight, Time %in% c(0, 20))
tmp$WeightTertile <- cut(tmp$weight,
  breaks = quantile(tmp$weight, c(0, 1/3, 2/3, 1), na.rm = TRUE),
  include.lowest = TRUE)

egltable(c("weight", "WeightTertile"), g = "Time",
  data = tmp,
  idvar = "Chick", paired = TRUE)
```

```
rm(tmp)
```

`empirical_pvalue` *Calculates an empirical p-value based on the data*

Description

This function takes a vector of statistics and calculates the empirical p-value, that is, how many fall on the other side of zero. It calculates a two-tailed p-value.

Usage

```
empirical_pvalue(x, na.rm = TRUE)
```

Arguments

`x`	a data vector to operate on
`na.rm`	Logical whether to remove NA values. Defaults to `TRUE`

Value

a named vector with the number of values falling at or below zero, above zero, and the empirical p-value.

Author(s)

Joshua F. Wiley <josh@elkhartgroup.com>

Examples

```
empirical_pvalue(rnorm(100))
```

`findSigRegions` *Function to find significant regions from an interaction*

Description

This function uses the `contrast` function from **rms** to find the threshold for significance from interactions.

Usage

```
findSigRegions(
  object,
  l1,
  l2,
  name.vary,
  lower,
  upper,
  alpha = 0.05,
  starts = 50
)
```

Arguments

object	A fitted rms object
l1	the first set of values to fix for the contrast function
l2	the second set of values to fix for the contrast function
name.vary	the name of the model parameter to vary values for to find the threshold. Note that this should not be included in l1 or l2 arguments.
lower	The lower bound to search for values for the varying value
upper	The upper bound to search for values for the varying value
alpha	The significance threshold, defaults to .05
starts	Number of starting values to try between the lower and upper bounds.

Value

A data table with notes if no convergence or significance thresholds (if any).

Examples

```
## make me
```

---

formatHtest *Function to format the reuslts of a hypothesis test as text*

Description

Function to format the reuslts of a hypothesis test as text

Usage

```
formatHtest(
  x,
  type = c("t", "F", "chisq", "kw", "mh", "r_pearson", "r_kendall", "r_spearman"),
  ...
)
```

Arguments

x	A htest class object
type	The type of htest. Currently one of: “t”, “F”, “chisq”, “kw”, “mh”, “r_pearson”, “r_kendall”, or “r_spearman” for t-tests, F-tests, chi-square tests, kruskal-wallis tests, Mantel-Haenszel tests, pearson correlations, kendall tau correlation, and spearman rho correlation, respectively.
...	Arguments passed on to p-value formatting

Value

A character string with results

Examples

```
formatHtest(t.test(extra ~ group, data = sleep), type = "t")
formatHtest(anova(aov(mpg ~ factor(cyl), data = mtcars)), type = "F")
formatHtest(chisq.test(c(A = 20, B = 15, C = 25)), type = "chisq")
formatHtest(kruskal.test(Ozone ~ Month, data = airquality), type = "kw")
formatHtest(mantelhaen.test(UCBAdmissions), type = "mh")
formatHtest(cor.test(~ mpg + hp, data = mtcars, method = "pearson"), type = "r_pearson")
formatHtest(cor.test(~ mpg + hp, data = mtcars, method = "kendall"), type = "r_kendall")
formatHtest(cor.test(~ mpg + hp, data = mtcars, method = "spearman"), type = "r_spearman")
```

---

formatMedIQR *Function to format the median and IQR of a variable*

Description

Function to format the median and IQR of a variable

Usage

```
formatMedIQR(x, d = 2, na.rm = TRUE)
```

Arguments

x	the data to have the median and IQR calculated
d	How many digits to display. Defaults to 2.
na.rm	Logical whether to remove missing values. Defaults to TRUE.

Value

A character string with results

Examples

```
formatMedIQR(mtcars$mpg)
```

formatPval *Function to simplify formatting p-values for easy viewing / publication*

Description

Function to simplify formatting p-values for easy viewing / publication

Usage

```
formatPval(
  x,
  d = 3,
  sd,
  includeP = FALSE,
  includeSign = FALSE,
  dropLeadingZero = TRUE
)
```

Arguments

x	p values to convert
d	number of digits
sd	number of scientific digits. Defaults to d if missing.
includeP	logical value whether to include the character "p" itself. Defaults to FALSE.
includeSign	logical value whether to include the character "=" or "<". Defaults to FALSE and if includeP = TRUE it must be TRUE.
dropLeadingZero	logical value whether to drop leading zeros for p-values. Defaults to TRUE.

Value

A character string with stars

Examples

```
formatPval(c(.00052456, .000000124, .01035, .030489, .534946))
formatPval(c(.00052456, .000000124, .01035, .030489, .534946), 3, 3, FALSE, TRUE)
formatPval(c(.00052456, .000000124, .01035, .030489, .534946), 3, 3, TRUE, TRUE)
formatPval(c(.00052456, .000000124, .01035, .030489, .534946), 5)
formatPval(c(1, .15346, .085463, .05673, .04837, .015353462,
  .0089, .00164, .0006589, .0000000053326), 3, 5)
formatPval(c(1, .15346, .085463, .05673, .04837, .015353462,
  .0089, .00164, .0006589, .0000000053326), 3, 5, dropLeadingZero = FALSE)
```

`geom_tufterange` *Tufte Range*

Description

Make axis lines informative by making them show the observed range of the data. Inspired from the excellent ggthemes package: https://github.com/jrnold/ggthemes

Usage

```
geom_tufterange(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`stat`	The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following: • A `Stat` ggproto subclass, for example `StatCount`. • A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`. • For more information and other ways to specify the stat, see the layer stat documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

... — Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat's documentation lists which parameters it can accept.
- The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

---

gglikert *Creates a plot for likert scale*

Description

Creates a plot for likert scale

Usage

```
gglikert(
  x,
  y,
  leftLab,
  rightLab,
  colour,
  data,
  xlim,
  title,
  shape = 18,
  size = 7
)
```

Arguments

x	Variable to plot on the x axis (the likert scale responses or averages)
y	The variable containing an index of the different items, should be integers
leftLab	The variable with anchors for the low end of the Likert scale
rightLab	The variable with anchors for the high end of the Likert scale
colour	A character string giving the name of a variable for colouring the data, like a grouping variable. Alternately the colour of points passed to geom_point
data	The data to use for plotting
xlim	A vector giving the lower an upper limit for the x axis. This should be the possible range of the Likert scale, not the actual range.
title	A character vector giving the title for the plot
shape	A number indicating the point shape, passed to geom_point
size	A number indicating the size of points, passed to geom_point

Examples

```
library(JWileymisc)
library(ggplot2)
library(ggpubr)
testdat <- data.table::data.table(
  Var = 1:4,
  Mean = c(1.5, 3, 2.2, 4.6),
  Low = c("Happy", "Peaceful", "Excited", "Content"),
  High = c("Sad", "Angry", "Hopeless", "Anxious"))

gglikert("Mean", "Var", "Low", "High", data = testdat, xlim = c(1, 5),
  title = "Example Plot of Average Affect Ratings")

testdat <- rbind(
  cbind(testdat, Group = "Young"),
  cbind(testdat, Group = "Old"))
testdat$Mean[5:8] <- c(1.7, 2.6, 2.0, 4.4)
```

```
gglikert("Mean", "Var", "Low", "High", colour = "Group",
  data = testdat, xlim = c(1, 5),
  title = "Example Plot of Average Affect Ratings")

gglikert("Mean", "Var", "Low", "High", colour = "Group",
  data = testdat, xlim = c(1, 5),
  title = "Example Plot of Average Affect Ratings") +
ggplot2::scale_colour_manual(values = c("Young" = "grey50", "Old" = "black"))

## clean up
rm(testdat)
```

`hashDataset` *Create a character vector or file hash of a dataset and each variable*

Description

Given a `data.frame` or `data.table`, create a character vector MD5 hash of the overall dataset and each variable. The goal of this is to create a secure vector / text file that can be tracked using version control (e.g., GitHub) without requiring commiting sensitive datasets. The tracking will make it possible to evaluate whether two datasets are the same, such as when sending data or when datasets may change over time to know which variable(s) changed, if any.

Usage

```
hashDataset(x, file)
```

Arguments

`x`	A `data.frame` or `data.table` to be hashed.
`file`	An optional character string. If given, assumed to be the path/name of a file to write the character string hash out to, for convenience. When non missing, the character vector is returned invisibly and a file written. When missing (default), the character vector is returned directly.

Value

A (possibly invisible) character vector. Also (optionally) a text file written version of the character string.

Examples

```
hashDataset(mtcars)

## if a file is specified it will write the results to the text file
## nicely formatted, along these lines

cat(hashDataset(cars), sep = "\n")
```

`intSigRegGraph` *Function to find significant regions from an interaction*

Description

This function uses the `contrast` function from **rms** to find the threshold for significance from interactions.

Usage

```
intSigRegGraph(
  object,
  predList,
  contrastList,
  xvar,
  varyvar,
  varyvar.levels,
  xlab = xvar,
  ylab = "Predicted Values",
  ratio = 1,
  xlim,
  ylim,
  xbreaks,
  xlabels = xbreaks,
  scale.x = c(m = 0, s = 1),
  scale.y = c(m = 0, s = 1),
  starts = 50
)
```

Arguments

`object`	A fitted rms object
`predList`	TODO
`contrastList`	TODO
`xvar`	TODO
`varyvar`	TODO
`varyvar.levels`	TODO
`xlab`	optional
`ylab`	TODO
`ratio`	TODO
`xlim`	TODO
`ylim`	TODO
`xbreaks`	TODO

xlabels	optional
scale.x	optional
scale.y	optional
starts	Number of starting values to try between the lower and upper bounds.

Value

A data table with notes if no convergence or significance thresholds (if any).

Examples

```
## make me
```

---

is.naz *Is a variable missing, non finite or zero length character?*

Description

Given a vector, return TRUE or FALSE if each element is either missing (NA/NaN), non finite (e.g. infinite) or a zero length character string (only for character vectors).

Usage

```
is.naz(x)
```

Arguments

x	A vector to identify missing / non finite or zero length strings from

Value

a logical vector

Examples

```
is.naz(c(1, NA, NaN))
is.naz(c(1, NA, NaN, Inf))
is.naz(c("test", "", NA_character_))
```

lagk *Create a lagged variable*

Description

Given a variable, create a k lagged version, optionally do it by a grouping factor, such as an ID.

Usage

```
lagk(x, k = 1, by)
```

Arguments

x	the variable to lag
k	the length to lag it
by	a variable to lag by. Must be sorted.

Value

a vector of the lagged values

Examples

```
lagk(1:4, 1)
```

lm2 *Modified lm() to use a specified design matrix*

Description

This function is a minor modification of the lm() function to allow the use of a pre-specified design matrix. It is not intended for public use but only to support `modelTest.lm`.

Usage

```
lm2(
  formula,
  data,
  subset,
  weights,
  na.action,
  model = TRUE,
  x = FALSE,
  y = FALSE,
  qr = TRUE,
```

```
  singular.ok = TRUE,
  contrasts = NULL,
  offset,
  designMatrix,
  yObserved,
  ...
)
```

Arguments

formula	An object of class "formula" although it is only minimally used
data	the dataset
subset	subset
weights	any weights
na.action	Defaults to na.omit
model	defaults to TRUE
x	defaults to FALSE
y	defaults to FALSE
qr	defaults to TRUE
singular.ok	defaults to TRUE
contrasts	defaults to NULL
offset	missing by default
designMatrix	a model matrix / design matrix (all numeric, pre coded if applicable for discrete variables)
yObserved	the observed y values
...	additional arguments

Value

an lm class object

See Also

lm

Examples

```
mtcars$cyl <- factor(mtcars$cyl)
m <- lm(mpg ~ hp * cyl, data = mtcars)

x <- model.matrix(m)
y <- mtcars$mpg
m2 <- JWileymisc:::lm2(mpg ~ 1 + cyl + hp:cyl, data = mtcars,
  designMatrix = x[, -2, drop = FALSE],
  yObserved = y)
```

```
anova(m, m2)

rm(m, m2, x, y)
```

`meanCircular` *Calculate a Circular Mean*

Description

Function to calculate circular mean

Usage

```
meanCircular(x, max, na.rm = TRUE)
```

Arguments

`x`	Numeric or integer values
`max`	The theoretical maximum (e.g., if degrees, 360)
`na.rm`	A logical value indicating whether to remove missing values. Defaults to `TRUE`.

Value

A numeric value with the circular mean.

Examples

```
meanCircular(c(22:23, 1:2), max = 24)
meanCircular(c(12, 24), max = 24)
meanCircular(c(6, 7, 23), max = 24)
meanCircular(c(6, 7, 21), max = 24)
meanCircular(c(6, 21), max = 24)
meanCircular(c(6, 23), max = 24)
meanCircular(c(.91, .96, .05, .16), max = 1)
meanCircular(c(6, 7, 8, 9), max = 24)
meanCircular(1:3, max = 24)
meanCircular(21:23, max = 24)
meanCircular(c(16, 17, 18, 19), max = 24)
meanCircular(c(355, 5, 15), max = 360)
```

`modelCompare` *Compare Two Models*

Description

Generic function.

Usage

```
modelCompare(model1, model2, ...)

as.modelCompare(x)

is.modelCompare(x)

## S3 method for class 'lm'
modelCompare(model1, model2, ...)
```

Arguments

`model1`	A fitted model object.
`model2`	A fitted model object to compare to `model1`
`...`	Additional arguments passed to specific methods.
`x`	An object (e.g., list or a modelCompare object) to test or attempt coercing to a modelCompare object.

Value

Depends on the method dispatch.

Examples

```
m1 <- lm(mpg ~ qsec * hp, data = mtcars)

m2 <- lm(mpg ~ am, data = mtcars)

modelCompare(m1, m2)

## cleanup
rm(m1, m2)

## Not run:
m3 <- lm(mpg ~ 1, data = mtcars)
m4 <- lm(mpg ~ 0, data = mtcars)
modelCompare(m3, m4)

## cleanup
rm(m3, m4)
```

```
## End(Not run)
```

`modelDiagnostics` *Model Diagnostics Functions*

Description

A set of functions to calculate model diagnostics on models, including constructors, a generic function, a test of whether an object is of the `modelDiagnostics` class, and methods.

Usage

```
modelDiagnostics(object, ...)

as.modelDiagnostics(x)

is.modelDiagnostics(x)

## S3 method for class 'lm'
modelDiagnostics(
  object,
  ev.perc = 0.001,
  robust = FALSE,
  distr = "normal",
  standardized = TRUE,
  ...
)
```

Arguments

`object`	A fitted model object, with methods for `model.frame`, `resid` and `fitted`.
`...`	Additional arguments, passed to methods or `residualDiagnostics`.
`x`	An object to test or a list to coerce to a `modelDiagnostics` object.
`ev.perc`	A real number between 0 and 1 indicating the proportion of the theoretical distribution beyond which values are considered extreme values (possible outliers). Defaults to .001.
`robust`	Whether to use robust mean and standard deviation estimates for normal distribution
`distr`	A character string given the assumed distribution. Passed on to `testDistribution`. Defaults to "normal".
`standardized`	A logical whether to use standardized residuals. Defaults to `TRUE` generally where possible but may depend on method.

Value

A logical (`is.modelDiagnostics`) or a modelDiagnostics object (list) for `as.modelDiagnostics` and `modelDiagnostics`.

Examples

```
testm <- stats::lm(mpg ~ hp * factor(cyl), data = mtcars)

md <- modelDiagnostics(testm)
plot(md$residualDiagnostics$testDistribution)
md$extremeValues

plot(md)

md <- modelDiagnostics(testm, ev.perc = .1)
md$extremeValues
plot(md, ncol = 2)

testdat <- data.frame(
  y = c(1, 2, 2, 3, 3, NA, 9000000, 2, 2, 1),
  x = c(1, 2, 3, 4, 5, 6, 5, 4, 3, 2))

modelDiagnostics(
  lm(y ~ x, data = testdat, na.action = "na.omit"),
  ev.perc = .1)$extremeValues

modelDiagnostics(
  lm(y ~ x, data = testdat, na.action = "na.exclude"),
  ev.perc = .1)$extremeValues

## clean up
rm(testm, md, testdat)
```

---

modelPerformance *Return Indices of Model Performance*

Description

Generic function. Generally returns things like fit indices, absolute error metrics, tests of overall model significance.

Usage

```
modelPerformance(object, ...)

as.modelPerformance(x)

is.modelPerformance(x)
```

```
## S3 method for class 'lm'
modelPerformance(object, ...)
```

Arguments

object	A fitted model object. The class of the model determines which specific method is called.
...	Additional arguments passed to specific methods.
x	A object (e.g., list or a modelPerformance object) to test or attempt coercing to a modelPerformance object.

Details

For lm class objects, return number of observations, AIC, BIC, log likelihood, R2, overall model F test, and p-value.

Value

A data.table with results.

A list with a data.table with the following elements:

Model A character string indicating the model type, here lm

N_Obs The number of observations

AIC Akaike Information Criterion

BIC Bayesian Information Criterion

LL log likelihood

LLDF log likelihood degrees of freedom

Sigma Residual variability

R2 in sample variance explained

F2 Cohen's F2 effect size R2 / (1 - R2)

AdjR2 adjusted variance explained

F F value for overall model significance test

FNumDF numerator degrees of freedom for F test

FDenDF denominator degrees of freedom for F test

P p-value for overall model F test

Examples

```
modelPerformance(lm(mpg ~ qsec * hp, data = mtcars))

modelPerformance(lm(mpg ~ hp, data = mtcars))

## Not run:
modelPerformance(lm(mpg ~ 0 + hp, data = mtcars))
modelPerformance(lm(mpg ~ 1, data = mtcars))
modelPerformance(lm(mpg ~ 0, data = mtcars))

## End(Not run)
```

`modelTest` *Detailed Tests on Models*

Description

TODO: make me!

Usage

```
modelTest(object, ...)

is.modelTest(x)

as.modelTest(x)

## S3 method for class 'vglm'
modelTest(object, ...)

## S3 method for class 'lm'
modelTest(object, ...)
```

Arguments

`object`	A fitted model object.
`...`	Additional arguments passed to specific methods.
`x`	A object (e.g., list or a modelTest object) to test or attempt coercing to a modelTest object.

Value

Depends on the method dispatch.

A list with two elements. `Results` contains a data table of the actual estimates. `Table` contains a nicely formatted character matrix.

A list with two elements. `Results` contains a data table of the actual estimates. `Table` contains a nicely formatted character matrix.

Examples

```
mtcars$cyl <- factor(mtcars$cyl)
m <- VGAM::vglm(cyl ~ qsec,
  family = VGAM::multinomial(), data = mtcars)
modelTest(m)

## clean up
rm(m, mtcars)

## Not run:
```

```
mtcars$cyl <- factor(mtcars$cyl)
mtcars$am <- factor(mtcars$am)
m <- VGAM::vglm(cyl ~ qsec,
  family = VGAM::multinomial(), data = mtcars)
modelTest(m)

m <- VGAM::vglm(cyl ~ scale(qsec),
  family = VGAM::multinomial(), data = mtcars)
modelTest(m)

m2 <- VGAM::vglm(cyl ~ factor(vs) * scale(qsec),
  family = VGAM::multinomial(), data = mtcars)
modelTest(m2)

m <- VGAM::vglm(Species ~ Sepal.Length,
  family = VGAM::multinomial(), data = iris)
modelTest(m)

set.seed(1234)
sampdata <- data.frame(
  Outcome = factor(sample(letters[1:3], 20 * 9, TRUE)),
  C1 = rnorm(20 * 9),
  D3 = sample(paste0("L", 1:3), 20 * 9, TRUE))

m <- VGAM::vglm(Outcome ~ factor(D3),
  family = VGAM::multinomial(), data = sampdata)
modelTest(m)

m <- VGAM::vglm(Outcome ~ factor(D3) + C1,
  family = VGAM::multinomial(), data = sampdata)
modelTest(m)

## End(Not run)
m1 <- lm(mpg ~ qsec * hp, data = mtcars)
modelTest(m1)

mtcars$cyl <- factor(mtcars$cyl)
m2 <- lm(mpg ~ cyl, data = mtcars)
modelTest(m2)

m3 <- lm(mpg ~ hp * cyl, data = mtcars)
modelTest(m3)

m4 <- lm(sqrt(mpg) ~ hp * cyl, data = mtcars)
modelTest(m4)

m5 <- lm(mpg ~ sqrt(hp) * cyl, data = mtcars)
modelTest(m5)

## cleanup
rm(m1, m2, m3, m4, m5, mtcars)
```

`moments` *Estimate the first and second moments*

Description

This function relies on the **lavaan** package to use the Expectation Maximization (EM) algorithm to estimate the first and second moments (means and [co]variances) when there is missing data.

Usage

```
moments(data, ...)
```

Arguments

`data`	A data frame or an object coercable to a data frame. The means and covariances of all variables are estimated.
`...`	Additional arguments passed on to the `estimate.moments.EM` function in **lavaan**. Note this is not an exported function.

Value

A list containing the esimates from the EM algorithm.

`mu`	A named vector of the means.
`sigma`	The covariance matrix.

Author(s)

Suggested by Yves Rosseel author of the lavaan package on which this depends

See Also

`SEMSummary`

Examples

```
# sample data
Xmiss <- as.matrix(iris[, -5])
# make 25% missing completely at random
set.seed(10)
Xmiss[sample(length(Xmiss), length(Xmiss) * .25)] <- NA
Xmiss <- as.data.frame(Xmiss)

# true means and covariance
colMeans(iris[, -5])
# covariance with n - 1 divisor
cov(iris[, -5])

# means and covariance matrix using list wise deletion
```

```
colMeans(na.omit(Xmiss))
cov(na.omit(Xmiss))

# means and covariance matrix using EM
moments(Xmiss)
# clean up
rm(Xmiss)
```

---

naz.omit *Missing and Zero Character Omit*

Description

Given a vector, exclude any missing values, not a number values, non finite values, and if a character class, any zero length strings.

Usage

```
naz.omit(x)
```

Arguments

x	A vector to exclude missing, non finite or zero length strings from

Value

a vector with missing/non finite/zero length strings omitted

Examples

```
## stats na.omit
stats::na.omit(c(1, NA, NaN))
stats::na.omit(c("test", "", NA_character_))

naz.omit(c(1, NA, NaN))
naz.omit(c(1L, NA))
naz.omit(c(1L, NA, Inf))
naz.omit(c("test", "", NA_character_))
```

param_summary *Calculates summaries for a parameter*

Description

This function takes a vector of statistics and calculates several summaries: mean, median, 95 the empirical p-value, that is, how many fall on the other side of zero.

Usage

```
param_summary(x, trans = function(x) x, ..., na.rm = TRUE)
```

Arguments

x	a data vector to operate on
trans	A function to transform the data. Used for summaries, but not p-values. Defaults to the identity function.
...	Additional arguments passed to formatPval to control p-value printing.
na.rm	Logical whether to remove NA values. Defaults to TRUE

Value

A data frame of summary statistics

Examples

```
param_summary(rnorm(100))
```

param_summary_format *Format a data frame of summary statistics*

Description

This functions nicely formats a data frame of parameter summary statistics and is designed to be used with the param_summary() function.

Usage

```
param_summary_format(d, digits = getOption("digits"), pretty = FALSE)
```

Arguments

d	A data frame of the parameter summary statistics
digits	Number of digits to round to for printing
pretty	Logical value whether prettified values should be returned. Defaults to FALSE.

Value

A formatted data.table of summary statistics or a formated vector (if `pretty = TRUE`).

Examples

```
set.seed(1234)
xsum <- do.call(rbind, apply(matrix(rnorm(100*10), ncol = 10),
  2, param_summary))
rownames(xsum) <- letters[1:10]
param_summary_format(xsum)
param_summary_format(xsum, pretty = TRUE)

rm(xsum)
```

---

plot.modelDiagnostics.lm

Plot Diagnostics for an lm model

Description

This function creates a number of diagnostic plots from lm models.

Usage

```
## S3 method for class 'modelDiagnostics.lm'
plot(x, y, plot = TRUE, ask = TRUE, ncol, ...)
```

Arguments

`x`	A `modelDiagnostics` class object from lm.
`y`	Included to match the generic. Not used.
`plot`	A logical value whether or not to plot the results or simply return the graaphical objects.
`ask`	A logical whether to ask before changing plots. Only applies to interactive environments.
`ncol`	The number of columns to use for plots. Missing by default which means individual plots are created. If specified, plots are put together in a grid.
`...`	Included to match the generic. Not used.

Value

a list including plots of the residuals, residuals versus fitted values

Examples

```
testm <- stats::lm(mpg ~ hp * factor(cyl), data = mtcars)

md <- modelDiagnostics(testm, ev.perc = .1)

plot(md)
plot(md, ncol = 2)

## clean up
rm(testm, md)
```

`plot.residualDiagnostics` *Plot Residual Diagnostics Default Method*

Description

This function creates a number of diagnostic plots from residuals. It is a default method.

Usage

```
## S3 method for class 'residualDiagnostics'
plot(x, y, plot = TRUE, ask = TRUE, ncol, ...)
```

Arguments

`x`	A `residualDiagnostics` class object.
`y`	Included to match the generic. Not used.
`plot`	A logical value whether or not to plot the results or simply return the graphical objects.
`ask`	A logical whether to ask before changing plots. Only applies to interactive environments.
`ncol`	The number of columns to use for plots. Missing by default which means individual plots are created. If specified, plots are put together in a grid.
`...`	Included to match the generic. Not used.

Value

a list including plots of the residuals, residuals versus fitted values

Examples

```
testm <- stats::lm(mpg ~ hp * factor(cyl), data = mtcars)
testm <- stats::lm(mpg ~ factor(cyl), data = mtcars)

md <- residualDiagnostics(testm, ev.perc = .1)

plot(md, plot = FALSE)$ResFittedPlot
plot(md, ncol = 2)

## clean up
rm(testm, md)
```

`plot.SEMSummary` *Plots SEMSummary object*

Description

Plots SEMSummary object

Usage

```
## S3 method for class 'SEMSummary'
plot(x, y, ...)
```

Arguments

`x`	An object of class SEMSummary.
`y`	Ignored
`...`	Additional arguments passed on to the real workhorse, `corplot`.

See Also

`corplot`, `SEMSummary`

Examples

```
# default plot
plot(SEMSummary(~ ., data = mtcars))

# same as default
plot(SEMSummary(~ ., data = mtcars), type = "coverage")

# shows p values
plot(SEMSummary(~ ., data = mtcars), type = "p")

# shows correlations
plot(SEMSummary(~ ., data = mtcars), type = "cor")
```

plot.SEMSummary.list *Plots SEMSummary.list object*

Description

Plots SEMSummary.list object

Usage

```
## S3 method for class 'SEMSummary.list'
plot(x, y, which, plot = TRUE, ...)
```

Arguments

x	An object of class SEMSummary.list.
y	Ignored
which	either a numeric vector based on the positions, or a character vector giving the names of the levels of the list to plot.
plot	A logical, whether to actually plot the results or not. Defaults to TRUE.
...	Additional arguments passed on to the real workhorse, corplot.

See Also

corplot, SEMSummary

Examples

```
## correlation matrix by am level
plot(SEMSummary(~ . | am, data = mtcars))
```

plot.testDistribution *Plot method for testDistribution objects*

Description

Make plots of testDistribution objects, including density and QQ plots.

Usage

```
## S3 method for class 'testDistribution'
plot(
  x,
  y,
  xlim = NULL,
  varlab = "X",
  plot = TRUE,
  rugthreshold = 500,
  seed = 1234,
  factor = 1,
  ...
)
```

Arguments

x	A list of class "testDistribution".
y	Included to match the generic. Not used.
xlim	An optional vector to control the x limits for the theoretical distribution density line, useful when densities become extreme at boundary values to help keep the scale of the graph reasonable. Passed on to stat_function.
varlab	A character vector the label to use for the variable
plot	A logical vector whether to plot the graphs. Defaults to TRUE.
rugthreshold	Integer determining the number of observations beyond which no rug plot is added. Note that even if this threshold is exceeded, a rug plot will still be added for any extreme values (if extreme values are used and present).
seed	a random seed used to make the jitter added for Poisson and Negative Binomial distributions reproducible
factor	A scale factor fo the amount of jitter added to the QQ and Deviates plots for Poisson and Negative Binomial distributions. Defaults to 1. This results in 1 * smallest distance between points / 5 being used.
...	Additional arguments.

Value

An invisible list with the ggplot2 objects for graphs, as well as information about the distribution (parameter estimates, name, log likelihood (useful for comparing the fit of different distributions to the data), and a dataset with the sorted data and theoretical quantiles.

See Also

testDistribution

Examples

```
## evaluate mpg against a normal distribution
plot(testDistribution(mtcars$mpg))

## Not run:

## example data
set.seed(1234)
d <- data.table::data.table(
  Ynorm = rnorm(200),
  Ybeta = rbeta(200, 1, 4),
  Ychisq = rchisq(200, 8),
  Yf = rf(200, 5, 10),
  Ygamma = rgamma(200, 2, 2),
  Ynbinom = rnbinom(200, mu = 4, size = 9),
  Ypois = rpois(200, 4))

## testing and graphing
plot(testDistribution(d$Ybeta, "beta", starts = list(shape1 = 1, shape2 = 4)))
plot(testDistribution(d$Ychisq, "chisq", starts = list(df = 8)))

## for chi-square distribution, extreme values only on
## the right tail
plot(testDistribution(d$Ychisq, "chisq", starts = list(df = 8),
  extremevalues = "empirical", ev.perc = .1))
plot(testDistribution(d$Ychisq, "chisq", starts = list(df = 8),
  extremevalues = "theoretical", ev.perc = .1))

plot(testDistribution(d$Yf, "f", starts = list(df1 = 5, df2 = 10)))
plot(testDistribution(d$Ygamma, "gamma"))
plot(testDistribution(d$Ynbinom, "poisson"))
plot(testDistribution(d$Ynbinom, "nbinom"))
plot(testDistribution(d$Ypois, "poisson"))

## compare log likelihood of two different distributions
testDistribution(d$Ygamma, "normal")$Distribution$LL
testDistribution(d$Ygamma, "gamma")$Distribution$LL

plot(testDistribution(d$Ynorm, "normal"))
plot(testDistribution(c(d$Ynorm, 10, 1000), "normal",
  extremevalues = "theoretical"))
plot(testDistribution(c(d$Ynorm, 10, 1000), "normal",
  extremevalues = "theoretical", robust = TRUE))

plot(testDistribution(mtcars, "mvnormal"))

## for multivariate normal mahalanobis distance
## which follows a chi-square distribution, extreme values only on
## the right tail
plot(testDistribution(mtcars, "mvnormal", extremevalues = "empirical",
  ev.perc = .1))
plot(testDistribution(mtcars, "mvnormal", extremevalues = "theoretical",
```

```
  ev.perc = .1))

rm(d) ## cleanup

## End(Not run)
```

R2 *Calculate R2 Values*

Description

Generic function to return variance explained (R2) estimates from various models. In some cases these will be true R2 values, in other cases they may be pseudo-R2 values if R2 is not strictly defined for a model.

Usage

```
R2(object, ...)

## S3 method for class 'lm'
R2(object, ...)
```

Arguments

object	A fitted model object.
...	Additional arguments passed to specific methods.

Value

Depends on the method dispatch.

The raw and adjusted r-squared value.

Examples

```
R2(lm(mpg ~ qsec * hp, data = mtcars))
```

`residualDiagnostics` *Residual Diagnostics Functions*

Description

A set of functions to calculate residual diagnostics on models, including constructors, a generic function, a test of whether an object is of the `residualDiagnostics` class, and methods.

Usage

```
residualDiagnostics(object, ...)

as.residualDiagnostics(x)

is.residualDiagnostics(x)

## S3 method for class 'lm'
residualDiagnostics(
  object,
  ev.perc = 0.001,
  robust = FALSE,
  distr = "normal",
  standardized = TRUE,
  cut = 8L,
  quantiles = TRUE,
  ...
)
```

Arguments

`object`	A fitted model object, with methods for `model.frame`, `resid` and `fitted`.
`...`	Additional arguments passed to methods.
`x`	A object (e.g., list or a modelDiagnostics object) to test or attempt coercing to a residualDiagnostics object.
`ev.perc`	A real number between 0 and 1 indicating the proportion of the theoretical distribution beyond which values are considered extreme values (possible outliers). Defaults to .001.
`robust`	Whether to use robust mean and standard deviation estimates for normal distribution
`distr`	A character string given the assumed distribution. Passed on to `testDistribution`. Defaults to "normal".
`standardized`	A logical whether to use standardized residuals. Defaults to `TRUE` generally where possible but may depend on method.
`cut`	An integer, how many unique predicted values there have to be at least for predicted values to be treated continuously, otherwise they are treated as discrete values. Defaults to 8.

quantiles A logical whether to calculate quantiles for the residuals. Defaults to TRUE. If FALSE, then do not calculate them. These are based on simple quantiles for each predicted value if the predicted values are few enough to be treated discretely. See cut argument. Otherwise they are based on quantile regression. First trying smoothing splines, and falling back to linear quantil regression if the splines fail. You may also want to turn these off if they are not working well, or are not of value in your diagnostics.

Value

A logical (is.residualDiagnostics) or a residualDiagnostics object (list) for as.residualDiagnostics and residualDiagnostics.

Examples

```
testm <- stats::lm(mpg ~ hp * factor(cyl), data = mtcars)

resm <- residualDiagnostics(testm)
plot(resm$testDistribution)

resm <- residualDiagnostics(testm, standardized = FALSE)
plot(resm$testDistribution)

## clean up
rm(testm, resm)
## Not run:

testdat <- data.frame(
  y = c(1, 2, 2, 3, 3, NA, 9000000, 2, 2, 1),
  x = c(1, 2, 3, 4, 5, 6, 5, 4, 3, 2))

residualDiagnostics(
  lm(y ~ x, data = testdat, na.action = "na.omit"),
  ev.perc = .1)$Residuals

residualDiagnostics(
  lm(y ~ x, data = testdat, na.action = "na.exclude"),
  ev.perc = .1)$Residuals

residualDiagnostics(
  lm(sqrt(mpg) ~ hp, data = mtcars, na.action = "na.omit"),
  ev.perc = .1)$Residuals

## End(Not run)
```

---

roundedfivenum *Calculate a rounded five number summary*

Description

Numbers are the minimum, 25th percentile, median, 75th percentile, and maximum, of the non missing data. Values returned are either the significant digits or rounded values, whichever ends up resulting in the fewest total digits.

Usage

```
roundedfivenum(x, round = 2, sig = 3)
```

Arguments

x	The data to have the summary calculated on
round	The number of digits to try rounding
sig	The number of significant digits to try

Value

The rounded or significant digit five number summary

Examples

```
JWileymisc:::roundedfivenum(rnorm(1000))
JWileymisc:::roundedfivenum(mtcars$hp)
```

SEMSummary *Summary Statistics for a SEM Analysis*

Description

This function is designed to calculate the descriptive statistics and summaries that are often reported on raw data when the main analyses use structural equation modelling.

Usage

```
SEMSummary(
  formula,
  data,
  use = c("fiml", "pairwise.complete.obs", "complete.obs")
)
```

Arguments

formula	A formula of the variables to be used in the analysis. See the 'details' section for more information.
data	A data frame, matrix, or list containing the variables used in the formula. This is a required argument.
use	A character vector of how to handle missing data. Defaults to "fiml".

Details

This function calculates a variety of relevant statistics on the raw data used in a SEM analysis. Because it is meant for SEM style data, for now it expects all variables to be numeric. In the future I may try to expand it to handle factor variables somehow.

Both the formula and data arguments are required. The formula should be the right hand side only. The most common way to use it would be with variable names separated by '+s'. For convenience, a '.' is expanded to mean "all variables in the data set". For a large number of variables or when whole datasets are being analyzed, this can be considerably easier to write. Also it facilitates column indexing by simply passing a subset of the data (e.g., `data[, 1:10]`) and using the '.' expansion to analyze the first 10 columns. The examples section demonstrate this use.

Also noteworthy is that `SEMSummary` is not really meant to be used on its own. It is the computational workhorse, but it is meant to be used with a styling or printing method to produce simple output. `APAStyler` has methods for `SEMSummary` output.

There are several new ways to handle missing data now including listwise deletion, pairwise deletion, and using the EM algorithm, the default.

Value

A list with S3 class "SEMSummary"

`names`	A character vector containing the variable names.
`n`	An integer vector of the length of each variable used (this includes available and missing data).
`nmissing`	An integer vector of the number of missing values in each variable.
`mu`	A vector of the arithmetic means of each variable (on complete data).
`stdev`	A numeric vector of the standard deviations of each variable (on complete data).
`Sigma`	The numeric covariance matrix for all variables.
`sSigma`	The numeric correlation matrix for all variables.
`coverage`	A numeric matrix giving the percentage (technically decimal) of information available for each pairwise covariance/correlation.
`pvalue`	The two-sided p values for the correlation matrix. Pairwise present N used to calculate degrees of freedom.

See Also

`APAStyler`

Examples

```
## Example using the built in iris dataset
s <- SEMSummary(~ Sepal.Length + Sepal.Width + Petal.Length, data = iris)
s # show output ... not very nice

## Prettier output from SEMSummary
APAStyler(s)
```

```
#### Subset the dataset and use the . expansion ####

## summary for all variables in mtcars data set
## with 11 variables, this could be a pain to write out
SEMSummary(~ ., data = mtcars)

## . expansion is also useful when we know column positions
## but not necessarily names
SEMSummary(~ ., data = mtcars[, c(1, 2, 3, 9, 10, 11)])

## clean up
rm(s)

## sample data
Xmiss <- as.matrix(iris[, -5])
# make q0% missing completely at random
set.seed(10)
Xmiss[sample(length(Xmiss), length(Xmiss) * .10)] <- NA
Xmiss <- as.data.frame(Xmiss)

SEMSummary(~ ., data = Xmiss, use = "fiml")


## pairwise
APAStyler(SEMSummary(~ ., data = Xmiss, use = "pair"),
  type = "cor")

## same as cor()
cor(Xmiss, use = "pairwise.complete.obs")

## complete cases only
SEMSummary(~ ., data = Xmiss, use = "comp")

## clean up
rm(Xmiss)
```

SEMSummary.fit *Summary Statistics for a SEM Analysis*

Description

This is a low level fitting function, for SEMSummary.

Usage

```
SEMSummary.fit(
  formula,
  data,
  use = c("fiml", "pairwise.complete.obs", "complete.obs")
)
```

Arguments

formula	A formula of the variables to be used in the analysis. See the 'details' section for more information.
data	A data frame, matrix, or list containing the variables used in the formula. This is a required argument.
use	A character vector of how to handle missing data. Defaults to "fiml".

Value

A list with S3 class "SEMSummary"

names	A character vector containing the variable names.
n	An integer vector of the length of each variable used (this includes available and missing data).
nmissing	An integer vector of the number of missing values in each variable.
mu	A vector of the arithmetic means of each variable (on complete data).
stdev	A numeric vector of the standard deviations of each variable (on complete data).
Sigma	The numeric covariance matrix for all variables.
sSigma	The numeric correlation matrix for all variables.
coverage	A numeric matrix giving the percentage (technically decimal) of information available for each pairwise covariance/correlation.
pvalue	The two-sided p values for the correlation matrix. Pairwise present N used to calculate degrees of freedom.

See Also

SEMSummary

---

smd *Calculate Standardized Mean Difference (SMD)*

Description

Simple function to calculate effect sizes for mean differences.

Usage

```
smd(x, g, index = c("all", "1", "2"))
```

Arguments

x	A continuous variable
g	A grouping variable, with two levels
index	A character string: "all" uses pooled variance, "1" uses the first factor level variance, "2" uses the second factor level variance.

Value

The standardized mean difference.

Examples

```
smd(mtcars$mpg, mtcars$am)
smd(mtcars$mpg, mtcars$am, "all")
smd(mtcars$mpg, mtcars$am, "1")
smd(mtcars$mpg, mtcars$am, "2")

smd(mtcars$hp, mtcars$vs)

d <- data.table::as.data.table(mtcars)
d[, smd(mpg, vs)]
rm(d)
```

---

star *Function to simplify converting p-values to asterisks*

Description

Function to simplify converting p-values to asterisks

Usage

```
star(x, includeMarginal = FALSE)
```

Arguments

x	p values to convert to stars
includeMarginal	logical value whether to include a symbol for marginally significant >.05 but < .10 p-values. Defaults to FALSE.

Value

A character string with stars

Examples

```
star(c(.0005, .001, .005, .01, .02, .05, .08, .1, .5, 1))
```

`testDistribution` *Test the distribution of a variable against a specific distribution*

Description

Function designed to help examine distributions. It also includes an option for assessing multivariate normality using the (squared) Mahalanobis distance. A generic function, some methods, and constructor (`as.testDistribution`) and function to check class (`is.testDistribution`) also are provided.

Note that for the `use` argument, several options are possible. By default it is "complete.obs", which uses only cases with complete data on all variables. Another option is "pairwise.complete.obs", which uses all available data for each variable indivdiually to estimate the means and variances, and all pairwise complete observation pairs for each covariance. Because the same cases are not used for all estimates, it is possible to obtain a covariance matrix that is not positive definite (e.g., correlations > +1 or < -1).

Finally, the last option is "fiml", which uses full information maximum likelihood estimates of the means and covariance matrix. Depending on the number of cases, missing data patterns, and variables, this may be quite slow and computationally demanding.

The `robust` argument determines whether to use robust estimates or not when calculating densities, etc. By default it is `FALSE`, but if `TRUE` and a univariate or multivariate normal distribution is tested, then robust estimates of the means and covariance matrix (a variance if univariate) will be used based on `covMcd` from the **robustbase** package.

Usage

```
testDistribution(x, ...)

as.testDistribution(x)

is.testDistribution(x)

## Default S3 method:
testDistribution(
  x,
 distr = c("normal", "beta", "chisq", "f", "gamma", "geometric", "nbinom", "poisson",
    "uniform", "mvnormal"),
  na.rm = TRUE,
  starts,
  extremevalues = c("no", "theoretical", "empirical"),
  ev.perc = 0.001,
  use = c("complete.obs", "pairwise.complete.obs", "fiml"),
  robust = FALSE,
  ...
)
```

Arguments

x	The data as a single variable or vector to check the distribution unless the distribution is "mvnormal" in which case it should be a data frame or data table.
...	Additional arguments. If these include mu and sigma and the distribution is multivariate normal, then it will use the passed values instead of calculating the mean and covariances of the data.
distr	A character string indicating the distribution to be tested. Currently one of: "normal", "beta", "chisq" (chi-squared), "f", "gamma", "geometric", "nbinom" (negative binomial), "poisson", "uniform", or "mvnormal" for multivariate normal where Mahalanobis distances are calculated and compared against a Chi-squared distribution with degrees of freedom equal to the number of variables.
na.rm	A logical value whether to omit missing values. Defaults to TRUE.
starts	A named list of the starting values. Not required for all distributions. Passed on to fitdistr which fits the maximum likelihood estimates of the distribution parameters.
extremevalues	A character vector whether to indicate extreme values. Should be "no" to do nothing, "empirical" to show extreme values based on the observed data percentiles, or "theoretical" to show extreme values based on percentiles of the theoretical distribution.
ev.perc	Percentile to use for extreme values. For example if .01, then the lowest 1 percent and highest 1 percent will be labelled extreme values. Defaults to the lowest and highest 0.1 percent.
use	A character vector indicating how the moments (means and covariance matrix) should be estimated in the presence of missing data when distr = mvnormal. The default is to use complete observations, but full information maximum likelihood based on functions in **lavaan** is also available. See details.
robust	A logical whether to use robust estimation or not. Currently only applies to normally distributed data (univariate or multivariate). Also, when robust = TRUE, only complete observations are used (i.e., use = "complete.obs"). See details.

Value

A logical whether or not an object is of class testDistribution or an object of the same class.

A list with information about the distribution (parameter estimates, name, log likelihood (useful for comparing the fit of different distributions to the data), and a dataset with the sorted data and theoretical quantiles.

See Also

SEMSummary

Examples

```
## example data
set.seed(1234)
d <- data.table::data.table(
```

```
  Ynorm = rnorm(200),
  Ybeta = rbeta(200, 1, 4),
  Ychisq = rchisq(200, 8),
  Yf = rf(200, 5, 10),
  Ygamma = rgamma(200, 2, 2),
  Ynbinom = rnbinom(200, mu = 4, size = 9),
  Ypois = rpois(200, 4))

## testing and graphing
testDistribution(d$Ybeta, "beta", starts = list(shape1 = 1, shape2 = 4))
testDistribution(d$Ychisq, "chisq", starts = list(df = 8))

## for chi-square distribution, extreme values only on
## the right tail
testDistribution(d$Ychisq, "chisq", starts = list(df = 8),
  extremevalues = "empirical", ev.perc = .1)
testDistribution(d$Ychisq, "chisq", starts = list(df = 8),
  extremevalues = "theoretical", ev.perc = .1)

## Not run:

testDistribution(d$Yf, "uniform")
testDistribution(d$Ypois, "geometric")

testDistribution(d$Yf, "f", starts = list(df1 = 5, df2 = 10))
testDistribution(d$Ygamma, "gamma")
testDistribution(d$Ynbinom, "poisson")
testDistribution(d$Ynbinom, "nbinom")
testDistribution(d$Ypois, "poisson")

## compare log likelihood of two different distributions
testDistribution(d$Ygamma, "normal")$Distribution$LL
testDistribution(d$Ygamma, "gamma")$Distribution$LL

testDistribution(d$Ynorm, "normal")
testDistribution(c(d$Ynorm, 10, 1000), "normal",
  extremevalues = "theoretical")
testDistribution(c(d$Ynorm, 10, 1000), "normal",
  extremevalues = "theoretical", robust = TRUE)

testDistribution(mtcars, "mvnormal")

## for multivariate normal mahalanobis distance
## which follows a chi-square distribution, extreme values only on
## the right tail
testDistribution(mtcars, "mvnormal", extremevalues = "empirical",
  ev.perc = .1)
testDistribution(mtcars, "mvnormal", extremevalues = "theoretical",
  ev.perc = .1)

rm(d) ## cleanup

## End(Not run)
```

theme_tufte *Tufte Inspired Theme*

Description

This is a barebones theme. It turns many aspects of plot background lines, etc. off completely. Inspired from the excellent ggthemes package: https://github.com/jrnold/ggthemes

Usage

```
theme_tufte(base_size = 11, base_family = "sans")
```

Arguments

base_size	base font size, given in pts.
base_family	base font family

timeshift *Shift a time variable to have a new center (zero point)*

Description

Given a vector, shift the values to have a new center, but keeping the same minimum and maximum. Designed to work with time values where the minimum indicates the same time as the maximum (e.g., 24:00:00 is the same as 00:00:00).

Usage

```
timeshift(x, center = 0, min = 0, max = 1, inverse = FALSE)
```

Arguments

x	the time scores to shift
center	A value (between the minimum and maximum) to center the time scores. Defaults to 0, which has no effect.
min	The theoretical minimum of the time scores. Defaults to 0.
max	the theoretical maximum of the time scores. Defaults to 1.
inverse	A logical value, whether to 'unshift' the time scores. Defaults to FALSE.

Value

A vector of shifted time scores, recentered as specified.

Examples

```
## example showing centering at 11am (i.e., 11am becomes new 0)
plot((1:24)/24, timeshift((1:24)/24, 11/24))

## example showing the inverse, note that 24/24 becomes 0
plot((1:24)/24, timeshift(timeshift((1:24)/24, 11/24), 11/24, inverse = TRUE))
```

TukeyHSDgg *Tukey HSD Plot*

Description

This calculates and displays means, confidence intervals as well as which groups are different based on Tukey's HSD. Inspired by http://stackoverflow.com/questions/18771516/is-there-a-function-to-add-aov-post-hoc-testing-results-to-ggplot2-boxplot

Usage

```
TukeyHSDgg(x, y, d, ci = 0.95, idvar, ...)
```

Arguments

x	A categorical grouping variable name.
y	A continuous outcome variable name.
d	A dataset
ci	A numeric value indicating the coverage of the confidence interval to use. Defaults to 0.95.
idvar	An optional ID variable for multilevel data
...	Additional arguments passed on.

Value

A ggplot graph object.

Examples

```
## examples using it with single level data
## differences based on an ANOVA and follow up contrasts
mtcars$cyl <- factor(mtcars$cyl)
TukeyHSDgg("cyl", "mpg", mtcars)
rm(mtcars)

## Not run:
TukeyHSDgg("Species", "Sepal.Length", iris)

## example based on multilevel data
## differences based on model fit with lmer and follow up contrasts
```

```
TukeyHSDgg("treatment", "decrease", OrchardSprays, idvar = "colpos")

## End(Not run)
```

VAConverter *Visual Acuity Converter*

Description

Converter character (string) input of Snellen fractions, Counting Fingers (CF), and Hand Motion (HM) to logMAR values for use in statistical models. Can handle linear interpolation if passed an appropriate chart or if the measures fit with the default chart.

Usage

```
VAConverter(
  OS,
  OD,
  chart.values = NULL,
  chart.nletters = NULL,
  datatype = c("snellen", "decimal", "logMAR"),
  zero = 3
)
```

Arguments

OS	The values to be converted for the left eye (oculus sinister).
OD	The values to be converted for the right eye (oculus dexter)
chart.values	The Snellen fractions for the chart used (if interpolation is necessary and it is different from the default).
chart.nletters	The number of letters on each line of the chart that was used. Necessary for proper interpolation.
datatype	The type of data passed to OS and OD. One of "Snellen" (the default), "decimal", or "logMAR". Determines what transformations are needed to convert to logMAR values.
zero	The "zero" logMAR value. This is used as the zero point for visual acuity. For example, for light perception (LP), no light perception (NLP), etc. It defaults to 3 (which is equivalent to a Snellen value of 20/20000), but may also be NA. See details.

Details

VAConverter is primarily designed to take raw character data of various forms and convert them to logMAR values. Acceptable examples include: "20/20", "20/80 + 3", "20/20 - 4", "10/20", "CF 10", "HM 2", "CF 4", "NLP", "LP", "", "CF", "HM", etc. For Snellen values, both parts should be present, and there should be a space between components; e.g., between fraction, +/- and number or

between CF and 10. Although I have attempted to make it as flexible and general as possible, there are still fairly rigid requirements so that it can parse a variety of text formats to numerical values. Optionally, it can also handle decimal values (i.e., the results of actually dividing a Snellen value 20/20 = 1).

`chart.values` and `chart.nletters` must be the same length. These are used to interpolate values such as "20/20 + 3" which is interpreted as reading all of the letters on the "20/20" line and "3" of the letters on the next best line (typically "20/15" but this can be chart dependent). The functions goes 3/n of the distance between the logMAR values for each line. This is why it is important to know the values for the chart *that was actually used*.

If datatype = "logMAR", the values passed to `OS` and `OD` are directly assigned to the `logMAROS` and `logMAROD` slots of a `"VAObject"` and an error is returned if that results in the creation of an invalid object (e.g., they are not numeric or not of equal length).

The `zero` argument is primarily included to facilitate calculating averages. For example, in some cases it may be nice to get a sense of an individual's "overall" or "average" logMAR value. Because on the logMAR scale, 0 is "20/20", an alternate number needs to be used. 3 was chosen as a rough default, but it is by no means necessarily the best choice. If you are not interested in computing an average between the left and right eyes within individuals, it makes sense to simply use `NA` rather than a crude "zero" approximation.

Value

An object of class `VAObject`. This includes the left and right eye logMAR values in slots `@logMAROS` and `@logMAROD` as well as additional information. More information can be found in the class documentation.

Examples

```
## sampdat <- c("HM 12", "20/20 + 3", "20/50", "CF", "HM",
##              "20/70 - 2", "LP", NA, "Prosthetic")
## tmp <- VAConverter(OS = sampdat, OD = rev(sampdat), datatype = "snellen")
```

VAObject-class *An S4 class to hold visual acuity data*

Description

A class to hold Visual Acuity data for the oculus sinister (OS; left eye) and oculus dexter (OD; right eye)

Usage

```
## S4 method for signature 'VAObject'
show(object)

## S4 method for signature 'VAObject,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]
```

```
## S4 method for signature 'VAObject'
print(x, ...)

## S4 method for signature 'VAObject'
show(object)

## S4 method for signature 'VAObject'
summary(object, weightbest = TRUE, w = c(0.75, 0.25))
```

Arguments

object	A VAObject class object
x	the object to subset
i	the rows to subset (optional)
j	the columns to subset (optional)
...	Additional arguments passed to lower functions
drop	should be missing
weightbest	Logical whether to upweight the best seeing eye. Defaults to TRUE.
w	A numeric vector of the weights, first for the best seeing then the worst seeing eye. Defaults to c(.75, .25).

Methods (by generic)

- show(VAObject): show method
- x[i: extract method
- print(VAObject): print method
- show(VAObject): show method
- summary(VAObject): summary method

Slots

originalOS the original visual acuity data for the left (ocular sinister) eye

originalOD the original visual acuity data for the right (ocular dexter) eye

logMAROS Logarithm of the minimum angle of resolution data for OS

logMAROD Logarithm of the minimum angle of resolution data for OD

chart.values the snellen values for each line of the chart used to measure visual acuity. Used for the linear interpolation in the case of partially correct line readings.

chart.nletters the number of letters on each line of the chart used to measure visual acuity. Used for the linear interpolation in the case of partially correct line readings (+2 is 2/4 of the way to the next line if there are four letters, but only 2/6 if there are six, etc.)

zero the logMAR value chosen to represent "zero" visual acuity when creating the combined logMAR values for both eyes or taking the arithmetic mean.

VASummaryObject-class *An S4 class to hold visual acuity summary data*

Description

A class designed to hold visual acuity summary data

Usage

```
## S4 method for signature 'VASummaryObject'
show(object)

## S4 method for signature 'VASummaryObject,missing'
plot(x, y, ...)
```

Arguments

object	The object to be shown
x	A VASummaryObject
y	Should be missing
...	Additional, unused arguments

Methods (by generic)

- show(VASummaryObject): show method
- plot(x = VASummaryObject, y = missing): plot method

Slots

logMAR.combined Numeric values of the combined logarithm of the minimum angle of resolution data for both eyes

snellen.combined the snellen values back transformed from the combined logMAR values

mean.logMAR average of the logarithm of the minimum angle of resolution data

mean.snellen average of the combined Snellen data

`winsorizor` *Winsorize at specified percentiles*

Description

Simple function winsorizes data at the specified percentile.

Usage

```
winsorizor(d, percentile, values, na.rm = TRUE)
```

Arguments

`d`	A vector, matrix, data frame, or data table to be winsorized
`percentile`	The percentile bounded by [0, 1] to winsorize data at. If a data frame or matrix is provided for the data, this should have the same length as the number of columns, or it will be repeated for all.
`values`	If values are specified, use these instead of calculating by percentiles. Should be a data frame with columns named "low", and "high". If a data frame or matrix is provided for the data, there should be as many rows for values to winsorize at as there are columns in the data.
`na.rm`	A logical whether to remove NAs.

Value

winsorized data. Attributes are included to list the exact values (for each variable, if a data frame or matrix) used to winsorize at the lower and upper ends.

Examples

```
dev.new(width = 10, height = 5)
par(mfrow = c(1, 2))
hist(as.vector(eurodist), main = "Eurodist")
hist(winsorizor(as.vector(eurodist), .05),
  main = "Eurodist with lower and upper\n5% winsorized")

library(data.table)
dat <- data.table(x = 1:5)
dat[, y := scale(1:5)]
winsorizor(dat$y, .01)

## make a copy of the data table
winsorizor(dat, .01)

winsorizor(mtcars, .01)

winsorizor(matrix(1:9, 3), .01)

rm(dat) # clean up
```

Index