Package ‘MKinfer’

October 22, 2020

Version 0.6

Date 2020-10-21

Title Inferential Statistics

Author Matthias Kohl [aut, cre] (<https://orcid.org/0000-0001-9514-8910>)

Maintainer Matthias Kohl <Matthias.Kohl@stamats.de>

Depends R(>= 3.5.0)

Imports stats, MKdescr, boot, arrangements, nlme, ggplot2

Suggests knitr, rmarkdown, Amelia, exactRankTests

VignetteBuilder knitr

Description Computation of various confidence intervals (Altman et al. (2000), ISBN:978-0-727-91375-3; Hedderich and Sachs (2018), ISBN:978-3-662-56657-2) including bootstrapped versions (Davison and Hinkley (1997), ISBN:978-0-511-80284-3) as well as Hsu (Hedderich and Sachs (2018), ISBN:978-3-662-56657-2), permutation (Janssen (1997), <doi:10.1016/S0167-7152(97)00043-6>), bootstrap (Davison and Hinkley (1997), ISBN:978-0-511-80284-3) and multiple imputation (Barnard and Rubin (1999), <doi:10.1093/biomet/86.4.948>) t-test. Graphical visualization by volcano plots.

License LGPL-3

URL http://www.stamats.de/

NeedsCompilation no

Repository CRAN

Date/Publication 2020-10-22 19:40:03 UTC

R topics documented:

`MKinfer-package` *Inferential Statistics.*

Description

Computation of various confidence intervals (Altman et al. (2000), ISBN:978-0-727-91375-3; Hedderich and Sachs (2018), ISBN:978-3-662-56657-2) including bootstrapped versions (Davison and Hinkley (1997), ISBN:978-0-511-80284-3) as well as Hsu (Hedderich and Sachs (2018), ISBN:978-3-662-56657-2), permutation (Janssen (1997), <doi:10.1016/S0167-7152(97)00043-6>), bootstrap (Davison and Hinkley (1997), ISBN:978-0-511-80284-3) and multiple imputation (Barnard and Rubin (1999), <doi:10.1093/biomet/86.4.948>) t-test. Graphical visualization by volcano plots.

Details

Package:	MKinfer
Type:	Package
Version:	0.6
Date:	2020-10-21
Depends:	R(>= 3.5.0)
Imports:	stats, MKdescr, boot, arrangements, nlme, ggplot2
Suggests:	Amelia, knitr, rmarkdown, exactRankTests
License:	LGPL-3
URL:	http://www.stamats.de/

library(MKinfer)

Author(s)

Matthias Kohl `http://www.stamats.de`

Maintainer: Matthias Kohl <matthias.kohl@stamats.de>

binomCI *Confidence Intervals for Binomial Proportions*

Description

This function can be used to compute confidence intervals for binomial proportions.

Usage

```
binomCI(x, n, conf.level = 0.95, method = "wilson", rand = 123,
        R = 9999, bootci.type = "all",
        alternative = c("two.sided", "less", "greater"))
```

Arguments

x	number of successes
n	number of trials
conf.level	confidence level
method	character string specifing which method to use; see details.
rand	seed for random number generator; see details.
R	number of bootstrap replicates.
bootci.type	type of bootstrap interval; see boot.ci.
alternative	a character string specifying one- or two-sided confidence intervals. Must be one of "two.sided" (default), "greater" or "less" (one-sided intervals). You can specify just the initial letter.

Details

The Wald interval is obtained by inverting the acceptance region of the Wald large-sample normal test. There is also a Wald interval with continuity correction ("wald-cc").

The Wilson interval, which is the default, was introduced by Wilson (1927) and is the inversion of the CLT approximation to the family of equal tail tests of p = p0. The Wilson interval is recommended by Agresti and Coull (1998) as well as by Brown et al (2001).

The Agresti-Coull interval was proposed by Agresti and Coull (1998) and is a slight modification of the Wilson interval. The Agresti-Coull intervals are never shorter than the Wilson intervals; cf. Brown et al (2001).

The Jeffreys interval is an implementation of the equal-tailed Jeffreys prior interval as given in Brown et al (2001).

The modified Wilson interval is a modification of the Wilson interval for x close to 0 or n as proposed by Brown et al (2001).

The modified Jeffreys interval is a modification of the Jeffreys interval for `x == 0 | x == 1` and `x == n-1 | x == n` as proposed by Brown et al (2001).

The Clopper-Pearson interval is based on quantiles of corresponding beta distributions. This is sometimes also called exact interval.

The arcsine interval is based on the variance stabilizing distribution for the binomial distribution.

The logit interval is obtained by inverting the Wald type interval for the log odds.

The Witting interval (cf. Beispiel 2.106 in Witting (1985)) uses randomization to obtain uniformly optimal lower and upper confidence bounds (cf. Satz 2.105 in Witting (1985)) for binomial proportions.

The bootstrap interval is calculated by using function `boot.ci`.

For more details we refer to Brown et al (2001) as well as Witting (1985).

Value

A list with class `"confint"` containing the following components:

`estimate` the estimated probability of success.

`conf.int` a confidence interval for the probability of success.

Note

A first version of this function appeared in R package SLmisc.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

A. Agresti and B.A. Coull (1998). Approximate is better than "exact" for interval estimation of binomial proportions. *American Statistician*, **52**, 119-126.

L.D. Brown, T.T. Cai and A. Dasgupta (2001). Interval estimation for a binomial proportion. *Statistical Science*, **16**(2), 101-133.

H. Witting (1985). *Mathematische Statistik I*. Stuttgart: Teubner.

See Also

`binom.test`, `binconf`

Examples

```
binomCI(x = 42, n = 43, method = "wald")
binomCI(x = 42, n = 43, method = "wald-cc")
binomCI(x = 42, n = 43, method = "wilson")
binomCI(x = 42, n = 43, method = "agresti-coull")
binomCI(x = 42, n = 43, method = "jeffreys")
binomCI(x = 42, n = 43, method = "modified wilson")
binomCI(x = 42, n = 43, method = "modified jeffreys")
binomCI(x = 42, n = 43, method = "clopper-pearson")
binomCI(x = 42, n = 43, method = "arcsine")
```

```
binomCI(x = 42, n = 43, method = "logit")
binomCI(x = 42, n = 43, method = "witting")
## bootstrap intervals (R = 999 to reduce computation time for R checks)
binomCI(x = 42, n = 43, method = "boot", R = 999) # may generate values > 1!

## the confidence interval computed by binom.test
## corresponds to the Clopper-Pearson interval
binomCI(x = 42, n = 43, method = "clopper-pearson")$conf.int
binom.test(x = 42, n = 43)$conf.int

## one-sided intervals
binomCI(x = 10, n = 43, alternative = "less")
binomCI(x = 10, n = 43, alternative = "less", method = "boot",
        bootci.type = "bca", R = 999)
binomCI(x = 10, n = 43, alternative = "greater", method = "boot",
        bootci.type = "perc", R = 999)
```

---

binomDiffCI *Confidence Intervals for Difference of Two Binomial Proportions*

Description

This function can be used to compute confidence intervals for the difference of two binomial proportions. It includes methods for the independent and the paired case.

Usage

```
binomDiffCI(a, b, c, d, conf.level = 0.95, paired = FALSE,
            method = ifelse(paired, "wilson-cc", "wilson"),
            R = 9999, bootci.type = "all",
            alternative = c("two.sided", "less", "greater"))
```

Arguments

a	independent: number of successes of group 1; paired: number of cases with success in group 1 and 2.
b	independent: number of successes of group 2; paired: number of cases with success in group 1 and failure in group 2.
c	independent: number of failures of group 1; paired: number of cases with failure in group 1 and success in group 2.
d	independent: number of failures of group 2; paired: number of cases with failure in group 1 and 2.
conf.level	confidence level
paired	a logical value indicating whether the two groups are paired.
method	character string specifing which method to use; see details.
R	number of bootstrap replicates.

bootci.type
: type of bootstrap interval; see boot.ci.

alternative
: a character string specifying one- or two-sided confidence intervals. Must be one of "two.sided" (default), "greater" or "less" (one-sided intervals). You can specify just the initial letter.

Details

The Wald intervals (independent and paired) are obtained by applying the normal approximation. There are also Wald intervals with continuity correction.

The Wilson intervals are recommended by Newcombe and Altman (2000); see Chapter 6 of Altman et al. (2000). In the paired case, the continuity corrected version of the interval is recommended. The intervals are proposed in Newcombe (1998a) and Newcombe (1998b).

The bootstrap interval is calculated by using function boot.ci.

Value

A list with class "confint" containing the following components:

estimate
: the estimated probability of success.

conf.int
: a confidence interval for the probability of success.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

D. Altman, D. Machin, T. Bryant, M. Gardner (eds). *Statistics with Confidence: Confidence Intervals and Statistical Guidelines*, 2nd edition. John Wiley and Sons 2000.

R.G. Newcombe (1998a). Interval estimation for the difference between independent proportions: comparison of eleven methods. *Stat Med*, **17**(8), 873-890.

R.G. Newcombe (1998b). Improved confidence intervals for the difference between binomial proportions based on paired data. *Stat Med*, **17**(22), 2635-2650.

See Also

prop.test, boot.ci

Examples

```
## Example 1: Altman et al. (2000, p. 49)
## the confidence interval computed by prop.test
prop.test(c(63, 38), c(93, 92))$conf.int
## wald / simple asymptotic interval
binomDiffCI(a = 63, b = 38, c = 30, d = 54, method = "wald")
## wald / simple asymptotic interval with continuity correction
binomDiffCI(a = 63, b = 38, c = 30, d = 54, method = "wald-cc")
## wilson
binomDiffCI(a = 63, b = 38, c = 30, d = 54)
```

```
## bootstrap intervals (R = 999 to reduce computation time for R checks)
binomDiffCI(a = 63, b = 38, c = 30, d = 54, method = "boot", R = 999)
## one-sided
binomDiffCI(a = 63, b = 38, c = 30, d = 54, alternative = "greater")
## bootstrap intervals (R = 999 to reduce computation time for R checks)
binomDiffCI(a = 63, b = 38, c = 30, d = 54, method = "boot", R = 999,
            bootci.type = "bca", alternative = "greater")


## Example 2: Altman et al. (2000, p. 50)
## the confidence interval computed by prop.test
prop.test(c(5, 0), c(56, 29))$conf.int
## wald / simple asymptotic interval
binomDiffCI(a = 5, b = 0, c = 51, d = 29, method = "wald")
## wald / simple asymptotic interval with continuity correction
binomDiffCI(a = 5, b = 0, c = 51, d = 29, method = "wald-cc")
## wilson
binomDiffCI(a = 5, b = 0, c = 51, d = 29)
## bootstrap intervals (R = 999 to reduce computation time for R checks)
binomDiffCI(a = 5, b = 0, c = 51, d = 29, method = "boot", R = 999)
## one-sided
binomDiffCI(a = 5, b = 0, c = 51, d = 29, alternative = "less")
## bootstrap intervals (R = 999 to reduce computation time for R checks)
binomDiffCI(a = 5, b = 0, c = 51, d = 29, method = "boot", R = 999,
            bootci.type = "perc", alternative = "less")


## Example 3: Altman et al. (2000, p. 51)
## wald / simple asymptotic interval
binomDiffCI(a = 14, b = 5, c = 0, d = 22, paired = TRUE, method = "wald")
## wald / simple asymptotic interval with continuity correction
binomDiffCI(a = 14, b = 5, c = 0, d = 22, paired = TRUE, method = "wald-cc")
## wilson
binomDiffCI(a = 14, b = 5, c = 0, d = 22, paired = TRUE, method = "wilson")
## wilson with continuity correction
binomDiffCI(a = 14, b = 5, c = 0, d = 22, paired = TRUE)
## bootstrap intervals (R = 999 to reduce computation time for R checks)
binomDiffCI(a = 14, b = 5, c = 0, d = 22, paired = TRUE, method = "boot", R = 999)

## Example 4: Altman et al. (2000, p. 51)
## wald / simple asymptotic interval
binomDiffCI(a = 212, b = 144, c = 256, d = 707, paired = TRUE, method = "wald")
## wald / simple asymptotic interval with continuity correction
binomDiffCI(a = 212, b = 144, c = 256, d = 707, paired = TRUE, method = "wald-cc")
## wilson
binomDiffCI(a = 212, b = 144, c = 256, d = 707, paired = TRUE, method = "wilson")
## wilson with continuity correction
binomDiffCI(a = 212, b = 144, c = 256, d = 707, paired = TRUE)
## bootstrap intervals (R = 999 to reduce computation time for R checks)
binomDiffCI(a = 212, b = 144, c = 256, d = 707, paired = TRUE, method = "boot",
        bootci.type = c("norm", "basic", "stud", "perc"), R = 999) ## type = "bca" gives error
```

boot.t.test *Bootstrap t-Test*

Description

Performs one and two sample bootstrap t-tests on vectors of data.

Usage

```
boot.t.test(x, ...)

## Default S3 method:
boot.t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, R = 9999, symmetric = FALSE, ...)

## S3 method for class 'formula'
boot.t.test(formula, data, subset, na.action, ...)
```

Arguments

x	a (non-empty) numeric vector of data values.
y	an optional (non-empty) numeric vector of data values.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
mu	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
paired	a logical indicating whether you want a paired t-test.
var.equal	a logical variable indicating whether to treat the two variances as being equal. If TRUE then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.
conf.level	confidence level of the interval.
R	number of bootstrap replicates.
symmetric	a logical variable indicating whether to assume symmetry in the two-sided test. If TRUE then the symmetric bootstrap p value otherwise the equal-tail boostrap p value is computed.
formula	a formula of the form lhs ~ rhs where lhs is a numeric variable giving the data values and rhs a factor with two levels giving the corresponding groups.
data	an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.

`na.action`	a function which indicates what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`.
`...`	further arguments to be passed to or from methods.

Details

The implemented test corresponds to the proposal of Chapter 16 of Efron and Tibshirani (1993).

The function returns bootstrapped p values and confidence intervals as well as the results ot the t-test without bootstrap.

The formula interface is only applicable for the 2-sample tests.

`alternative = "greater"` is the alternative that `x` has a larger mean than `y`.

If `paired` is `TRUE` then both `x` and `y` must be specified and they must be the same length. Missing values are silently removed (in pairs if `paired` is `TRUE`). If `var.equal` is `TRUE` then the pooled estimate of the variance is used. By default, if `var.equal` is `FALSE` then the variance is estimated separately for both groups and the Welch modification to the degrees of freedom is used.

If the input data are effectively constant (compared to the larger of the two means) an error is generated.

Value

A list with class `"boot.htest"` (derived from class `htest`) containing the following components:

`statistic`	the value of the t-statistic.
`parameter`	the degrees of freedom for the t-statistic.
`p.value`	the p-value for the test.
`boot.p.value`	the bootstrapped p-value for the test.
`conf.int`	a confidence interval for the mean appropriate to the specified alternative hypothesis.
`boot.conf.int`	a bootstrap percentile confidence interval for the mean appropriate to the specified alternative hypothesis.
`estimate`	the estimated mean or difference in means depending on whether it was a one-sample test or a two-sample test.
`null.value`	the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test.
`stderr`	the standard error of the mean (difference), used as denominator in the t-statistic formula.
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string indicating what type of t-test was performed.
`data.name`	a character string giving the name(s) of the data.

Note

Code and documentation are for large parts identical to function `t.test`.

References

B. Efron, R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC 1993.

See Also

t.test, meanCI, meanDiffCI, perm.t.test

Examples

```
require(graphics)

t.test(1:10, y = c(7:20))      # P = .00001855
boot.t.test(1:10, y = c(7:20))

t.test(1:10, y = c(7:20, 200)) # P = .1245    -- NOT significant anymore
boot.t.test(1:10, y = c(7:20, 200))


## Classical example: Student's sleep data
plot(extra ~ group, data = sleep)
## Traditional interface
with(sleep, t.test(extra[group == 1], extra[group == 2]))
with(sleep, boot.t.test(extra[group == 1], extra[group == 2]))
## Formula interface
t.test(extra ~ group, data = sleep)
boot.t.test(extra ~ group, data = sleep)
```

---

cvCI — *Confidence Intervals for Coefficient of Variation*

Description

This function can be used to compute confidence intervals for the (classical) coefficient of variation.

Usage

```
cvCI(x, conf.level = 0.95, method = "miller", R = 9999,
     bootci.type = c("norm", "basic", "perc", "bca"), na.rm = FALSE,
     alternative = c("two.sided", "less", "greater"))
```

Arguments

x	numeric vector with positive numbers.
conf.level	confidence level
method	character string specifing which method to use; see details.
R	number of bootstrap replicates; see details.
bootci.type	type of bootstrap interval; see boot.ci. Type "student" does not work.

`na.rm` logical. Should missing values be removed?

`alternative` a character string specifying one- or two-sided confidence intervals. Must be one of "two.sided" (default), "greater" or "less" (one-sided intervals). You can specify just the initial letter.

Details

For details about the confidence intervals we refer to Gulhar et al (2012) and Arachchige et al (2019).

In case of bootstrap intervals type "student" does not work, since no standard error of CV is provided.

Value

A list with class "confint" containing the following components:

`estimate` the estimated coefficient of variation.

`conf.int` a confidence interval for the coefficient of variation.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

C.N.P.G. Arachchige, L.A. Prendergast and R.G. Staudte (2019). Robust analogues to the Coefficient of Variation. https://arxiv.org/abs/1907.01110.

M. Gulhar, G. Kibria, A. Albatineh, N.U. Ahmed (2012). A comparison of some confidence intervals for estimating the population coefficient of variation: a simulation study. *Sort*, **36**(1), 45-69.

See Also

CV, boot.ci

Examples

```
x <- rnorm(100, mean = 10, sd = 2) # CV = 0.2
cvCI(x, method = "miller")
cvCI(x, method = "sharma")
cvCI(x, method = "curto")
cvCI(x, method = "mckay")
cvCI(x, method = "vangel")
cvCI(x, method = "panichkitkosolkul")
cvCI(x, method = "medmiller")
cvCI(x, method = "medmckay")
cvCI(x, method = "medvangel")
cvCI(x, method = "medcurto")
cvCI(x, method = "gulhar")
cvCI(x, method = "boot", R = 999) # R = 999 to reduce computation time for R checks
```

```
## one-sided
cvCI(x, alternative = "less")
cvCI(x, alternative = "greater")
cvCI(x, method = "boot", bootci.type = "bca", alternative = "less", R = 999)
```

`hsu.t.test` *Hsu Two-Sample t-Test*

Description

Performs Hsu two sample t-tests on vectors of data.

Usage

```
hsu.t.test(x, ...)

## Default S3 method:
hsu.t.test(x, y,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, conf.level = 0.95, ...)

## S3 method for class 'formula'
hsu.t.test(formula, data, subset, na.action, ...)
```

Arguments

`x`	a (non-empty) numeric vector of data values.
`y`	a (non-empty) numeric vector of data values.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.
`mu`	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
`conf.level`	confidence level of the interval.
`formula`	a formula of the form `lhs ~ rhs` where `lhs` is a numeric variable giving the data values and `rhs` a factor with two levels giving the corresponding groups.
`data`	an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used.
`na.action`	a function which indicates what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`.
`...`	further arguments to be passed to or from methods.

Details

The function and its documentation was adapted from `t.test`.

`alternative = "greater"` is the alternative that `x` has a larger mean than `y`.

If the input data are effectively constant (compared to the larger of the two means) an error is generated.

One should at least have six observations per group to apply the test; see Section 6.8.3 and 7.4.4.2 of Hedderich and Sachs (2018).

Value

A list with class `"htest"` containing the following components:

`statistic`	the value of the t-statistic.
`parameter`	the degrees of freedom for the t-statistic.
`p.value`	the p-value for the test.
`conf.int`	a confidence interval for the mean appropriate to the specified alternative hypothesis.
`estimate`	the estimated means and standard deviations.
`null.value`	the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test.
`stderr`	the standard error of the difference in means, used as denominator in the t-statistic formula.
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string indicating what type of t-test was performed.
`data.name`	a character string giving the name(s) of the data.

References

J. Hedderich, L. Sachs. *Angewandte Statistik: Methodensammlung mit R*. Springer 2018.

Hsu, P. (1938). Contribution to the theory of "student's" t-test as applied to the problem of two samples. *Statistical Research Memoirs* **2**, 1-24.

See Also

`t.test`

Examples

```
## Examples taken and adapted from function t.test
t.test(1:10, y = c(7:20))      # P = .00001855
t.test(1:10, y = c(7:20, 200)) # P = .1245    -- NOT significant anymore
hsu.t.test(1:10, y = c(7:20))
hsu.t.test(1:10, y = c(7:20, 200))

## Traditional interface
with(sleep, t.test(extra[group == 1], extra[group == 2]))
```

```
with(sleep, hsu.t.test(extra[group == 1], extra[group == 2]))
## Formula interface
t.test(extra ~ group, data = sleep)
hsu.t.test(extra ~ group, data = sleep)
```

`imputeSD` *Impute Standard Deviations for Changes from Baseline*

Description

The function imputes standard deviations for changes from baseline adopting the approach describe in the Cochrane handbook, Section 16.1.3.2.

Usage

```
imputeSD(SD1, SD2, SDchange)
```

Arguments

`SD1`	numeric vector, baseline SD.
`SD2`	numeric vector, follow-up SD.
`SDchange`	numeric vector, SD for changes from baseline.

Details

The function imputes standard deviations for changes from baseline adopting the approach describe in the Cochrane handbook (2019), Section 6.5.2.8.

1) Missing `SD1` are replaced by correspondig values of `SD2` and vice versa.

2) Correlations for complete data (rows) are computed.

3) Minimum, mean and maximum correlation (over rows) are computed.

4) Missing values of SDchange are computed by the formula provided in the handbook. The minimum, mean and maximum correlation are used leading to maximal, mean and minimal SD values that may be used for imputation as well as a sensitivity analysis.

Value

`data.frame` with possibly imputed SD1 and SD2 values as well as the given SDchange values are returen. Moreover, the computed correlations as well as possible values for the imputation of SDchange are returned.

Author(s)

Matthias Kohl `<Matthias.Kohl@stamats.de>`

References

Higgins JPT, Green S (editors). Cochrane Handbook for Systematic Reviews of Interventions Version 5.1.0 [updated March 2011]. The Cochrane Collaboration, 2011. Available from www.handbook.cochrane.org.

Examples

```
SD1 <- c(0.149, 0.022, 0.036, 0.085, 0.125, NA, 0.139, 0.124, 0.038)
SD2 <- c(NA, 0.039, 0.038, 0.087, 0.125, NA, 0.135, 0.126, 0.038)
SDchange <- c(NA, NA, NA, 0.026, 0.058, NA, NA, NA, NA)
imputeSD(SD1, SD2, SDchange)
```

---

mi.t.test *Multiple Imputation Student's t-Test*

Description

Performs one and two sample t-tests on multiple imputed datasets.

Usage

```
mi.t.test(miData, ...)

## Default S3 method:
mi.t.test(miData, x, y = NULL,
        alternative = c("two.sided", "less", "greater"), mu = 0,
        paired = FALSE, var.equal = FALSE, conf.level = 0.95,
        subset = NULL, ...)
```

Arguments

`miData`	list of multiple imputed datasets.
`x`	name of a variable that shall be tested.
`y`	an optional name of a variable that shall be tested (paired test) or a variable that shall be used to split into groups (unpaired test).
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.
`mu`	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
`paired`	a logical indicating whether you want a paired t-test.
`var.equal`	a logical variable indicating whether to treat the two variances as being equal. If `TRUE` then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.
`conf.level`	confidence level of the interval.
`subset`	an optional vector specifying a subset of observations to be used.
`...`	further arguments to be passed to or from methods.

Details

alternative = "greater" is the alternative that x has a larger mean than y.

If paired is TRUE then both x and y must be specified and they must be the same length. Missing values are not allowed as they should have been imputed. If var.equal is TRUE then the pooled estimate of the variance is used. By default, if var.equal is FALSE then the variance is estimated separately for both groups and the Welch modification to the degrees of freedom is used.

We use the approach of Rubin (1987) in combination with the adjustment of Barnard and Rubin (1999).

Value

A list with class "htest" containing the following components:

statistic	the value of the t-statistic.
parameter	the degrees of freedom for the t-statistic.
p.value	the p-value for the test.
conf.int	a confidence interval for the mean appropriate to the specified alternative hypothesis.
estimate	the estimated mean (one-sample test), difference in means (paired test), or estimated means (two-sample test) as well as the respective standard deviations.
null.value	the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test.
alternative	a character string describing the alternative hypothesis.
method	a character string indicating what type of t-test was performed.
data.name	a character string giving the name(s) of the data.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Rubin, D. (1987). *Multiple Imputation for Nonresponse in Surveys*. John Wiley \& Sons, New York.

Barnard, J. and Rubin, D. (1999). Small-Sample Degrees of Freedom with Multiple Imputation. *Biometrika*, **86**(4), 948-955.

See Also

t.test

Examples

```
## Generate some data
set.seed(123)
x <- rnorm(25, mean = 1)
x[sample(1:25, 5)] <- NA
y <- rnorm(20, mean = -1)
y[sample(1:20, 4)] <- NA
pair <- c(rnorm(25, mean = 1), rnorm(20, mean = -1))
g <- factor(c(rep("yes", 25), rep("no", 20)))
D <- data.frame(ID = 1:45, variable = c(x, y), pair = pair, group = g)

## Use Amelia to impute missing values
library(Amelia)
res <- amelia(D, m = 10, p2s = 0, idvars = "ID", noms = "group")

## Per protocol analysis (Welch two-sample t-test)
t.test(variable ~ group, data = D)
## Intention to treat analysis (Multiple Imputation Welch two-sample t-test)
mi.t.test(res$imputations, x = "variable", y = "group")

## Per protocol analysis (Two-sample t-test)
t.test(variable ~ group, data = D, var.equal = TRUE)
## Intention to treat analysis (Multiple Imputation two-sample t-test)
mi.t.test(res$imputations, x = "variable", y = "group", var.equal = TRUE)

## Specifying alternatives
mi.t.test(res$imputations, x = "variable", y = "group", alternative = "less")
mi.t.test(res$imputations, x = "variable", y = "group", alternative = "greater")

## One sample test
t.test(D$variable[D$group == "yes"])
mi.t.test(res$imputations, x = "variable", subset = D$group == "yes")
mi.t.test(res$imputations, x = "variable", mu = -1, subset = D$group == "yes",
          alternative = "less")
mi.t.test(res$imputations, x = "variable", mu = -1, subset = D$group == "yes",
          alternative = "greater")

## paired test
t.test(D$variable, D$pair, paired = TRUE)
mi.t.test(res$imputations, x = "variable", y = "pair", paired = TRUE)
```

---

normCI — *Confidence Intervals for Mean and Standard Deviation*

Description

This function can be used to compute confidence intervals for mean and standard deviation of a normal distribution.

Usage

```
normCI(x, mean = NULL, sd = NULL, conf.level = 0.95,
       boot = FALSE, R = 9999, bootci.type = "all", na.rm = TRUE,
       alternative = c("two.sided", "less", "greater"))
meanCI(x, conf.level = 0.95, boot = FALSE, R = 9999, bootci.type = "all",
       na.rm = TRUE, alternative = c("two.sided", "less", "greater"))
sdCI(x, conf.level = 0.95, boot = FALSE, R = 9999, bootci.type = "all",
     na.rm = TRUE, alternative = c("two.sided", "less", "greater"))
```

Arguments

`x`	vector of observations.
`mean`	mean if known otherwise `NULL`.
`sd`	standard deviation if known otherwise `NULL`.
`conf.level`	confidence level.
`boot`	a logical value indicating whether bootstrapped confidence intervals shall be computed.
`R`	number of bootstrap replicates.
`bootci.type`	type of bootstrap interval; see `boot.ci`.
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.
`alternative`	a character string specifying one- or two-sided confidence intervals. Must be one of "two.sided" (default), "greater" or "less" (one-sided intervals). You can specify just the initial letter.

Details

The standard confidence intervals for mean and standard deviation are computed that can be found in many textbooks, e.g. Chapter 4 in Altman et al. (2000).

In addition, bootstrap confidence intervals for mean and/or SD can be computed, where function `boot.ci` is applied.

Value

A list with class `"confint"` containing the following components:

`estimate`	the estimated mean and sd.
`conf.int`	confidence interval(s) for mean and/or sd.
`Infos`	additional information.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

D. Altman, D. Machin, T. Bryant, M. Gardner (eds). Statistics with Confidence: Confidence Intervals and Statistical Guidelines, 2nd edition 2000.

Examples

```
x <- rnorm(50)
## mean and sd unknown
normCI(x)
meanCI(x)
sdCI(x)

## one-sided
normCI(x, alternative = "less")
meanCI(x, alternative = "greater")
sdCI(x, alternative = "greater")

## bootstrap intervals (R = 999 to reduce computation time for R checks)
normCI(x, boot = TRUE, R = 999)
meanCI(x, boot = TRUE, R = 999)
sdCI(x, boot = TRUE, R = 999)

normCI(x, boot = TRUE, R = 999, alternative = "less")
meanCI(x, boot = TRUE, R = 999, alternative = "less")
sdCI(x, boot = TRUE, R = 999, alternative = "greater")

## sd known
normCI(x, sd = 1)
## bootstrap intervals only for mean (sd is ignored)
## (R = 999 to reduce computation time for R checks)
normCI(x, sd = 1, boot = TRUE, R = 999)

## mean known
normCI(x, mean = 0)
## bootstrap intervals only for sd (mean is ignored)
## (R = 999 to reduce computation time for R checks)
normCI(x, mean = 0, boot = TRUE, R = 999)
```

---

normDiffCI *Confidence Intervals for Difference of Means*

Description

This function can be used to compute confidence intervals for difference of means assuming normal distributions.

Usage

```
normDiffCI(x, y, conf.level = 0.95, paired = FALSE, method = "welch",
           boot = FALSE, R = 9999, bootci.type = "all", na.rm = TRUE,
           alternative = c("two.sided", "less", "greater"))
meanDiffCI(x, y, conf.level = 0.95, paired = FALSE, method = "welch",
           boot = FALSE, R = 9999, bootci.type = "all", na.rm = TRUE,
           alternative = c("two.sided", "less", "greater"))
```

Arguments

`x`	numeric vector of data values of group 1.
`y`	numeric vector of data values of group 2.
`conf.level`	confidence level.
`paired`	a logical value indicating whether the two groups are paired.
`method`	a character string specifing which method to use in the unpaired case; see details.
`boot`	a logical value indicating whether bootstrapped confidence intervals shall be computed.
`R`	number of bootstrap replicates.
`bootci.type`	type of bootstrap interval; see `boot.ci`.
`na.rm`	a logical value indicating whether `NA` values should be stripped before the computation proceeds.
`alternative`	a character string specifying one- or two-sided confidence intervals. Must be one of "two.sided" (default), "greater" or "less" (one-sided intervals). You can specify just the initial letter.

Details

The standard confidence intervals for the difference of means are computed that can be found in many textbooks, e.g. Chapter 4 in Altman et al. (2000).

The method `"classical"` assumes equal variances whereas methods `"welch"` and `"hsu"` allow for unequal variances. The latter two methods use different formulas for computing the degrees of freedom of the respective t-distribution providing the quantiles in the confidence interval. Instead of the Welch-Satterhwaite equation the method of Hsu uses the minimum of the group sample sizes minus 1; see Section 6.8.3 of Hedderich and Sachs (2018).

Value

A list with class `"confint"` containing the following components:

`estimate`	point estimate (mean of differences or difference in means).
`conf.int`	confidence interval.
`Infos`	additional information.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

D. Altman, D. Machin, T. Bryant, M. Gardner (eds). *Statistics with Confidence: Confidence Intervals and Statistical Guidelines*, 2nd edition. John Wiley and Sons 2000.

J. Hedderich, L. Sachs. *Angewandte Statistik: Methodensammlung mit R*. Springer 2018.

Examples

```
x <- rnorm(100)
y <- rnorm(100, sd = 2)
## paired
normDiffCI(x, y, paired = TRUE)
## (R = 999 to reduce computation time for R checks)
normDiffCI(x, y, paired = TRUE, boot = TRUE, R = 999)
## compare
normCI(x-y)
## (R = 999 to reduce computation time for R checks)
normCI(x-y, boot = TRUE, R = 999)

## unpaired
y <- rnorm(90, mean = 1, sd = 2)
## classical
normDiffCI(x, y, method = "classical")
## (R = 999 to reduce computation time for R checks)
normDiffCI(x, y, method = "classical", boot = TRUE, R = 999)
## Welch (default as in case of function t.test)
normDiffCI(x, y, method = "welch")
## (R = 999 to reduce computation time for R checks)
normDiffCI(x, y, method = "welch", boot = TRUE, R = 999)
## Hsu
normDiffCI(x, y, method = "hsu")
## In case of bootstrap there is no difference between welch and hsu
## (R = 999 to reduce computation time for R checks)
normDiffCI(x, y, method = "hsu", boot = TRUE, R = 999)

## one-sided
normDiffCI(x, y, alternative = "less")
normDiffCI(x, y, boot = TRUE, R = 999, alternative = "greater")


## Monte-Carlo simulation: coverage probability
M <- 100 # increase for more stable/realistic results!
CIhsu <- CIwelch <- CIclass <- matrix(NA, nrow = M, ncol = 2)
for(i in 1:M){
  x <- rnorm(10)
  y <- rnorm(30, sd = 0.1)
  CIclass[i,] <- normDiffCI(x, y, method = "classical")$conf.int
  CIwelch[i,] <- normDiffCI(x, y, method = "welch")$conf.int
  CIhsu[i,] <- normDiffCI(x, y, method = "hsu")$conf.int
}
## coverage probabilies
## classical
sum(CIclass[,1] < 0 & 0 < CIclass[,2])/M
```

```
## Welch
sum(CIwelch[,1] < 0 & 0 < CIwelch[,2])/M
## Hsu
sum(CIhsu[,1] < 0 & 0 < CIhsu[,2])/M
```

`pairwise.fun` *Compute pairwise values for a given function*

Description

The function computes pairwise values for a given function.

Usage

```
pairwise.fun(x, g, fun, ...)
```

Arguments

`x`	numeric vector.
`g`	grouping vector or factor
`fun`	some function where the first two arguments have to be numeric vectors for which the function computes some quantity; see example section below.
`...`	additional arguments to fun.

Details

The function computes pairwise values for a given function.

The implementation is in certain aspects analogously to `pairwise.t.test`.

Value

Vector with pairwise function values.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

`pairwise.t.test`

Examples

```
set.seed(13)
x <- rnorm(100)
g <- factor(sample(1:4, 100, replace = TRUE))
levels(g) <- c("a", "b", "c", "d")
pairwise.fun(x, g, fun = function(x, y) t.test(x,y)$p.value)
## in contrast to
pairwise.t.test(x, g, p.adjust.method = "none", pool.sd = FALSE)
```

`perm.t.test` *Permutation t-Test*

Description

Performs one and two sample permutation t-tests on vectors of data.

Usage

```
perm.t.test(x, ...)

## Default S3 method:
perm.t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, R = 9999, symmetric = TRUE, ...)

## S3 method for class 'formula'
perm.t.test(formula, data, subset, na.action, ...)
```

Arguments

`x`	a (non-empty) numeric vector of data values.
`y`	an optional (non-empty) numeric vector of data values.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.
`mu`	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
`paired`	a logical indicating whether you want a paired t-test.
`var.equal`	a logical variable indicating whether to treat the two variances as being equal. If `TRUE` then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.
`conf.level`	confidence level of the interval.
`R`	number of (Monte-Carlo) permutations.

`symmetric`	a logical variable indicating whether to assume symmetry in the two-sided test. If `TRUE` then the symmetric permutation p value otherwise the equal-tail permutation p value is computed.
`formula`	a formula of the form `lhs ~ rhs` where `lhs` is a numeric variable giving the data values and `rhs` a factor with two levels giving the corresponding groups.
`data`	an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used.
`na.action`	a function which indicates what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`.
`...`	further arguments to be passed to or from methods.

Details

The implemented test corresponds to the proposal of Chapter 15 of Efron and Tibshirani (1993) for equal variances as well as Janssen (1997) respectively Chung and Romano (2013) for unequal variances.

The function returns permutation p values and confidence intervals as well as the results ot the t-test without permutations.

The formula interface is only applicable for the 2-sample tests.

`alternative = "greater"` is the alternative that `x` has a larger mean than `y`.

If `paired` is `TRUE` then both `x` and `y` must be specified and they must be the same length. Missing values are silently removed (in pairs if `paired` is `TRUE`). If `var.equal` is `TRUE` then the pooled estimate of the variance is used. By default, if `var.equal` is `FALSE` then the variance is estimated separately for both groups and the Welch modification to the degrees of freedom is used.

If the input data are effectively constant (compared to the larger of the two means) an error is generated.

Value

A list with class `"perm.htest"` (derived from class `htest`) containing the following components:

`statistic`	the value of the t-statistic.
`parameter`	the degrees of freedom for the t-statistic.
`p.value`	the p-value for the test.
`perm.p.value`	the (Monte-Carlo) permutation p-value for the test.
`conf.int`	a confidence interval for the mean appropriate to the specified alternative hypothesis.
`perm.conf.int`	a (Monte-Carlo) permutation percentile confidence interval for the mean appropriate to the specified alternative hypothesis.
`estimate`	the estimated mean or difference in means depending on whether it was a one-sample test or a two-sample test.

null.value	the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test.
stderr	the standard error of the mean (difference), used as denominator in the t-statistic formula.
alternative	a character string describing the alternative hypothesis.
method	a character string indicating what type of t-test was performed.
data.name	a character string giving the name(s) of the data.

Note

Code and documentation are for large parts identical to function t.test.

References

B. Efron, R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC 1993.

A. Janssen (1997). Studentized permutation tests for non-i.i.d, hypotheses and the generalized Behrens-Fisher problem. *Statistics and Probability Letters*, **36**, 9-21.

E. Chung, J.P. Romano (2013). Exact and asymptotically robust permutation tests. *The Annals of Statistics*, **41**(2), 484-507.

See Also

t.test, meanCI, meanDiffCI, boot.t.test

Examples

```
require(graphics)

t.test(1:10, y = c(7:20))      # P = .00001855
perm.t.test(1:10, y = c(7:20))

t.test(1:10, y = c(7:20, 200)) # P = .1245    -- NOT significant anymore
perm.t.test(1:10, y = c(7:20, 200)) # perm.conf.int affected by outlier!


## Classical example: Student's sleep data
plot(extra ~ group, data = sleep)
## Traditional interface
with(sleep, t.test(extra[group == 1], extra[group == 2]))
with(sleep, perm.t.test(extra[group == 1], extra[group == 2]))
## Formula interface
t.test(extra ~ group, data = sleep)
perm.t.test(extra ~ group, data = sleep)
```

quantileCI *Confidence Intervals for Quantiles*

Description

These functions can be used to compute confidence intervals for quantiles (including median).

Usage

```
quantileCI(x, prob = 0.5, conf.level = 0.95, method = "exact",
           R = 9999, bootci.type = c("norm", "basic", "perc", "bca"),
           minLength = FALSE, na.rm = FALSE,
           alternative = c("two.sided", "less", "greater"))
medianCI(x, conf.level = 0.95, method = "exact",
         R = 9999, bootci.type = c("norm", "basic", "perc", "bca"),
         minLength = FALSE, na.rm = FALSE,
         alternative = c("two.sided", "less", "greater"))
madCI(x, conf.level = 0.95, method = "exact", minLength = FALSE,
      R = 9999, bootci.type = c("norm", "basic", "perc", "bca"),
      na.rm = FALSE, constant = 1.4826,
      alternative = c("two.sided", "less", "greater"))
```

Arguments

`x`	numeric data vector
`prob`	quantile
`conf.level`	confidence level
`method`	character string specifing which method to use; see details.
`minLength`	logical, see details
`R`	number of bootstrap replicates.
`bootci.type`	type of bootstrap interval; see `boot.ci`.
`na.rm`	logical, remove `NA` values.
`constant`	scale factor (see `mad`).
`alternative`	a character string specifying one- or two-sided confidence intervals. Must be one of "two.sided" (default), "greater" or "less" (one-sided intervals). You can specify just the initial letter.

Details

The exact confidence interval (`method = "exact"`) is computed using binomial probabilities; see Section 6.8.1 in Sachs and Hedderich (2009). If the result is not unique, i.e. there is more than one interval with coverage proability closest to `conf.level`, then a matrix of confidence intervals is returned. If `minLength = TRUE`, an exact confidence interval with minimum length is returned.

The asymptotic confidence interval (`method = "asymptotic"`) is based on the normal approximation of the binomial distribution; see Section 6.8.1 in Sachs and Hedderich (2009).

In case of discrete data, there are alternative bootstrap approaches that might give better results; see Jentsch and Leucht (2016).

Value

A list with components

`estimate`	the sample quantile.
`CI`	a confidence interval for the sample quantile.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

L. Sachs and J. Hedderich (2009). *Angewandte Statistik*. Springer.

C. Jentsch and A. Leucht (2016). Bootstrapping sample quantiles of discrete data. *Ann Inst Stat Math*, **68**: 491-539.

See Also

`binom.test`, `binconf`

Examples

```
## To get a non-trivial exact confidence interval for the median
## one needs at least 6 observations
set.seed(123)
x <- rnorm(8)
## exact confidence interval not unique
medianCI(x)
madCI(x)

## minimum length exact confidence interval
medianCI(x, minLength = TRUE)
madCI(x, minLength = TRUE)

## asymptotic confidence interval
medianCI(x, method = "asymptotic")
madCI(x, method = "asymptotic")

## bootstrap confidence interval
x <- rnorm(50)
medianCI(x)
medianCI(x, method = "asymptotic")
## (R = 999 to reduce computation time for R checks)
medianCI(x, method = "boot", R = 999)
```

```
madCI(x)
madCI(x, method = "asymptotic")
## (R = 999 to reduce computation time for R checks)
madCI(x, method = "boot", R = 999)

## confidence interval for quantiles
quantileCI(x, prob = 0.25)
quantileCI(x, prob = 0.25, method = "asymptotic")

quantileCI(x, prob = 0.75)
## (R = 999 to reduce computation time for R checks)
quantileCI(x, prob = 0.75, method = "boot", R = 999)

## one-sided
quantileCI(x, prob = 0.75, alternative = "greater")
medianCI(x, alternative = "less", method = "asymptotic")
madCI(x, alternative = "greater", method = "boot", R = 999)
```

`rm.oneway.test` *Test for Equal Means in a Repeated Measures One-Way Layout*

Description

Test whether two or more samples have the same locations in a repeated measures setting.

Usage

```
rm.oneway.test(x, g, id, method = "aov")
```

Arguments

`x`	numeric, response (outcome, dependent) variable.
`g`	factor, grouping (independent) variable.
`id`	factor, subject id (blocking variable).
`method`	name of method, possible methods are `"aov"`, `"lme"`, `"friedman"`, `"quade"`

Details

The function wraps the functions `aov`, `lme`, `friedman.test` and `quade.test` into one function for a repeated measures one-way layout.

Value

A list with class `"htest"` containing the following components:

`statistic`	the value of the test statistic.
`parameter`	the degrees of freedom of the exact or approximate F distribution of the test statistic.

p.value	the p-value of the test.
method	a character string indicating the test performed.
data.name	a character string giving the names of the data.

References

Chambers, J. M., Freeny, A and Heiberger, R. M. (1992), *Analysis of variance; designed experiments. Chapter 5 of Statistical Models in S*, eds J. M. Chambers and T. J. Hastie, Wadsworth \& Brooks/Cole.

Pinheiro, J.C., and Bates, D.M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer.

Myles Hollander and Douglas A. Wolfe (1973), *Nonparametric Statistical Methods*. New York: John Wiley \& Sons. Pages 139-146.

D. Quade (1979), Using weighted rankings in the analysis of complete blocks with additive block effects. *Journal of the American Statistical Association* **74**, 680-683.

William J. Conover (1999), *Practical nonparametric statistics*. New York: John Wiley \& Sons. Pages 373-380.

See Also

aov, lme, friedman.test, quade.test

Examples

```
set.seed(123)
outcome <- c(rnorm(10), rnorm(10, mean = 1.5), rnorm(10, mean = 1))
timepoints <- factor(rep(1:3, each = 10))
patients <- factor(rep(1:10, times = 3))
rm.oneway.test(outcome, timepoints, patients)
rm.oneway.test(outcome, timepoints, patients, method = "lme")
rm.oneway.test(outcome, timepoints, patients, method = "friedman")
rm.oneway.test(outcome, timepoints, patients, method = "quade")
```

---

volcano *Volcano Plots*

Description

Produce volcano plot(s) of the given effect size and p values.

Usage

```
volcano(x, ...)

## Default S3 method:
volcano(x, pval, effect0 = 0, sig.level = 0.05,
                              effect.low = NULL, effect.high = NULL,
                              color.low = "#4575B4", color.high = "#D73027",
                              xlab = "effect size", ylab = "-log10(p value)",
                              title = "Volcano Plot", alpha = 1, shape = 19, ...)
```

Arguments

`x`	in case of default method: measure of effect size.
`pval`	numeric, (adjusted) p values.
`effect0`	single numeric, value for no effect.
`sig.level`	single numeric, significance level.
`effect.low`	`NULL` or single numeric, boundary for low effect sizes.
`effect.high`	`NULL` or single numeric, boundary for low effect sizes.
`color.low`	color used if effect size smaller than `effect.low` and (adjusted) p value smaller than `sig.level`.
`color.high`	color used if effect size larger than `effect.high` and (adjusted) p value smaller than `sig.level`.
`xlab`	label of x-axis.
`ylab`	label of y-axis.
`title`	title of plot.
`alpha`	bleding factor (default: no blending.
`shape`	point shape used.
`...`	further arguments that may be passed through).

Details

The plot generates a `ggplot2` object that is shown.

Missing values are handled by the `ggplot2` functions.

Value

Object of class gg and `ggplot`.

Author(s)

Matthias Kohl <`Matthias.Kohl@stamats.de`>

References

Wikipedia contributors, *Volcano plot (statistics)*, Wikipedia, The Free Encyclopedia, `https://en.wikipedia.org/w/index.php?title=Volcano_plot_(statistics)&oldid=900217316` (accessed December 25, 2019).

For more sophisticated and flexible volcano plots see for instance: Blighe K, Rana S, Lewis M (2019). EnhancedVolcano: Publication-ready volcano plots with enhanced colouring and labeling. R/Bioconductor package. `https://github.com/kevinblighe/EnhancedVolcano`.

Examples

```
## Generate some data
x <- matrix(rnorm(1000, mean = 10), nrow = 10)
g1 <- rep("control", 10)
y1 <- matrix(rnorm(500, mean = 11.75), nrow = 10)
y2 <- matrix(rnorm(500, mean = 9.75, sd = 3), nrow = 10)
g2 <- rep("treatment", 10)
group <- factor(c(g1, g2))
Data <- rbind(x, cbind(y1, y2))
pvals <- apply(Data, 2, function(x, group) hsu.t.test(x ~ group)$p.value,
               group = group)
## compute log-fold change
logfc <- function(x, group){
  res <- tapply(x, group, mean)
  log2(res[1]/res[2])
}
lfcs <- apply(Data, 2, logfc, group = group)

volcano(lfcs, pvals, xlab = "log-fold change")
volcano(lfcs, pvals, effect.low = -0.25, effect.high = 0.25,
        xlab = "log-fold change")
volcano(lfcs, p.adjust(pvals, method = "fdr"),
        effect.low = -0.25, effect.high = 0.25,
        xlab = "log-fold change", ylab = "-log10(adj. p value)")
volcano(2^lfcs, pvals, effect0 = 1, effect.low = 1/2^0.25, effect.high = 2^0.25,
        xlab = "mean difference")
```

Index