Package 'MKinfer'

September 28, 2026

Version 1.4

Date 2026-09-27

Title Inferential Statistics

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

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

Depends R(>= 4.0.0)

Imports stats, utils, rlang, MKdescr, boot, arrangements, nlme, ggplot2, exactRankTests, miceadds, hypergeo, parallel

Suggests knitr, rmarkdown, Amelia, mice, mvtnorm

LazyData yes

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 Xiao (Xiao (2018), <doi:10.17654/TS054010021>), 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), intersection-union (Sozu et al. (2015), ISBN:978-3-319-22005-5) and multiple imputation (Barnard and Rubin (1999), <doi:10.1093/biomet/86.4.948>) t-test; furthermore, computation of intersection-union z-test as well as multiple imputation Wilcoxon tests. Graphical visualizations: volcano plot, Bland-Altman plots (Bland and Altman (1986), <doi:10.1016/S0140-6736(86)90837-8>; Shieh (2018), <doi:10.1186/s12874-018-0505-y>), mean difference plot (Boehning et al. (2008), <doi:10.1177/0962280207081867>), plot of test statistic for permutation and bootstrap tests as well as objects of class htest.

License LGPL-3

URL https://github.com/stamats/MKinfer

NeedsCompilation no

Repository CRAN

Date/Publication 2026-09-27 23:00:08 UTC

Contents

MKinfer-package . . . 2
baplot . . . 3
binomCI . . . 5
binomDiffCI . . . 8
boot.t.test . . . 10
cvCI . . . 13
dgt . . . 15
fingsys . . . 17
h0plot . . . 18
hsu.t.test . . . 20
imputeSD . . . 22
md2sens . . . 23
md2zfactor . . . 25
mdplot . . . 26
mi.t.test . . . 28
mi.wilcox.test . . . 30
mpe.t.test . . . 33
mpe.z.test . . . 35
normCI . . . 36
normDiffCI . . . 38
p2ses . . . 40
pairwise.ext.t.test . . . 42
pairwise.fun . . . 43
pairwise.wilcox.exact . . . 44
perm.t.test . . . 45
print.mpe.test . . . 49
quantileCI . . . 50
rm.oneway.test . . . 52
volcano . . . 54
xiao.t.test . . . 55

Index **58**

`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), intersection-union (Sozu et al. (2015), ISBN:978-3-319-22005-5) and multiple imputation (Barnard and Rubin (1999), <doi:10.1093/biomet/86.4.948>) t-test; furthermore, computation of intersection-union z-test as well as multiple imputation Wilcoxon tests. Graphical visualization by volcano and Bland-Altman plots (Bland and Altman (1986), <doi:10.1016/S0140-6736(86)90837-8>; Shieh (2018), <doi:10.1186/s12874-018-0505-y>).

Details

library(MKinfer)

Author(s)

Matthias Kohl https://www.stamats.de

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

`baplot` *Bland-Altman Plots*

Description

Produce classical parametric as well as nonparametric Bland-Altman plots including confidence intervals for the mean of the differences and the lower and upper limit of agreement.

Usage

```
baplot(x, y, loa.level = 0.95, conf.level = 0.95,
       xlab = "Mean", ylab = "Difference",
       title = "Bland-Altman Plot", xlim = NULL, ylim = NULL,
       type = c("parametric", "nonparametric"),
       loa.type = c("unbiased", "biased"), ci.diff = TRUE, ci.loa = TRUE,
       ci.type = c("exact", "approximate", "boot"),
       bootci.type = NULL, R = 9999, print.res = TRUE,
       color.low = "#4575B4", color.upp = "#D73027",
       alpha = 1, shape = 19, na.rm = TRUE, ...)
```

Arguments

`x`	numeric, measurements of method 1.
`y`	numeric, measurements of method 2.
`loa.level`	single numeric, level of the level of agreement.
`conf.level`	single numeric, significance level.
`xlab`	label of x-axis.
`ylab`	label of y-axis.
`title`	title of plot.
`xlim`	optional numeric vector of length 2, limits of x-axis.
`ylim`	optional numeric vector of length 2, limits of y-axis.
`type`	single character, either `"parametric"` (default) or `"nonparametric"`; see also Details below.
`loa.type`	single character, either `"unbiased"` (default) or `"biased"`; see also Details below.

ci.diff	single logical, plot confidence interval for the mean of the differences.
ci.loa	single logical, plot confidence intervals for the lower and upper limit of agreement.
ci.type	single character, either "exact" (default) or "approximate" or "boot"; see also Details below.
bootci.type	single character, type of bootstrap interval; see boot.ci and also Details below.
R	single numeric, number of bootstrap replicates.
print.res	single logical, print results of computations in addition to plotting.
color.low	single color (character), color for lower limit of agreement.
color.upp	single color (character), color for upper limit of agreement.
alpha	blending factor (default: no blending.
shape	point shape used.
na.rm	single logical, remove NA values before plotting.
...	further arguments passed to function boot, e.g. for parallel computing.

Details

The plot generates a ggplot2 object that is shown.

Setting type = "parametric" (default), a classical Bland-Altman plot with mean of the differences and standard lower and upper limit of agreement (mean +/- 1.96 SD) will be generated (loa.level = 0.95); see Altman and Bland (1983,1986).

Setting type = "nonparametric", a nonparametric Bland-Altman plot with median of the differences and empirical 2.5% and 97.5% quantiles (loa.level = 0.95) as lower and upper limit of agreement will be generated.

By changing loa.level the lower and upper limit of agreement will correspond to the (1-loa.level)/2 and the 1 - (1-low.level)/2 quantile.

Setting loa.type = "unbiased", the unbiased estimator of the standard deviation will be used; see Shieh (2018).

Setting loa.type = "biased", the standard deviation will be estimated by function sd.

Setting ci.type = "exact", the exact confidence intervals of Shieh (2018) will be used for the lower and upper limit of agreement.

Setting ci.type = "approximate", the approximate confidence intervals of Bland and Altman (1986, 1999) will be used.

Setting ci.type = "boot", bootstrap confidence intervals will be used. In the case of parametric Bland-Altman plots, the studentized bootstrap method will be used by default. In the case of nonparametric Bland-Altman plots, the bootstrap percentile method will be used by default.

The argument bootci.type can be set to "stud" (studentized bootstrap), "perc" (bootstrap percentile) or "bca" (adjusted bootstrap percentile).

Value

Object of class gg and ggplot.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Altman DG, Bland JM (1983). Measurement in Medicine: the Analysis of Method Comparison Studies. *The Statistician* 32:307-317.

Bland JM, Altman DG (1986). Statistical methods for assessing agreement between two methods of clinical measurement. *Lancet* 327(8476):307-10

Bland JM, Altman DG (1999). Measuring agreement in method comparison studies. *Stat Methods Med Res* 8:135-160.

Shieh G (2018). The appropriateness of Bland-Altman's approximate confidence intervals for limits of agreement. *BMC Med Res Methodol* 18:45.

Examples

```
data(fingsys)
## classical parametric Bland-Altman plot with exact confidence intervals
baplot(x = fingsys$armsys, y = fingsys$fingsys,
       xlab = "Mean of arm and finger [mm Hg]",
       ylab = "Difference (arm - finger) [mm Hg]")

## nonparametric Bland-Altman plot with exact confidence intervals
baplot(x = fingsys$armsys, y = fingsys$fingsys, type = "nonparametric",
       xlab = "Mean of arm and finger [mm Hg]",
       ylab = "Difference (arm - finger) [mm Hg]",
       title = "Nonparametric Bland-Altman Plot")
```

---

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.
...	further arguments passed to function boot, e.g. for parallel computing.

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

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)


## parallel computing for bootstrap
binomCI(x = 10, n = 43, method = "boot", R = 9999,
        parallel = "multicore", ncpus = 2)
```

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.
...	further arguments passed to function boot, e.g. for parallel computing.

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


binomDiffCI(a = 63, b = 38, c = 30, d = 54, method = "boot", R = 9999,
        parallel = "multicore", ncpus = 2)
```

`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,
       bootStat = 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 NAs. Defaults to getOption("na.action").
bootStat	logical; should the bootstrap results for the test statistic be returned. If TRUE, function h0plot can be used to plot the distribution of the results.
...	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).

Janssen and Pauls (2003) state in Example 5 (c) about Bootstrap two-sample tests: "From the previous comments it is clear that permutation tests should always be preferred for two-sample problems."

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.
`boot.estimate`	bootstrapped estimate.
`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.
`boot.stderr`	bootstrapped standard error.
`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, T. Pauls. How do bootstrap and permutation tests work?. *Ann. Statist.*, **31**(3), 768-806.

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))


## Traditional interface
with(mtcars, t.test(mpg[am == 0], mpg[am == 1]))
with(mtcars, boot.t.test(mpg[am == 0], mpg[am == 1]))
## Formula interface
t.test(mpg ~ am, data = mtcars)
boot.t.test(mpg ~ am, data = mtcars)
```

`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.
...	further arguments passed to function boot, e.g. for parallel computing.

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)


## parallel computing for bootstrap
cvCI(x, method = "boot", R = 9999, parallel = "multicore", ncpus = 2)
```

---

dgt *A Generalized Central t Distribution*

Description

Density, distribution function and quantile function for the generalized central t distribution defined in Section 5.3 of Xiao (2018)).

Usage

```
dgt(x, n1, n2, v1tov2, log = FALSE, MAX = 1e4)
pgt(q, n1, n2, v1tov2, lower.tail = TRUE, log.p = FALSE,
    rel.tol = .Machine$double.eps^0.25, parallel = FALSE, cl = NULL)
qgt(p, n1, n2, v1tov2, lower.tail = TRUE, log.p = FALSE,
    tol = .Machine$double.eps^0.5, parallel = FALSE, cl = NULL)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n1	sample size of first group.
n2	sample size of second group.
v1tov2	ratio of variances (var1/var2).
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
MAX	maximum number of iterations to determine the hypergeometric function; see function hypergeo.
rel.tol	relative accuracy of integrate.
tol	the desired accuracy (convergence tolerance) of uniroot.
parallel	logical; use parellel computing via function parRapply.
cl	a cluster object, created by package parallel or by package snow. If NULL, a new cluster will be generated with function makeCluster using the maximum number of available cores - 1.

Details

It is possible to provide arguments of length larger than 1 to the functions.

The density of the distribution can be found on page 58 of Xiao (2018). If the hypergeometric function does not converge, one can try to increase the value of `MAX`.

The distribution function is determined by numerical intergration via function `integrate`.

The quantile function is determined by applying function `uniroot`.

Value

`dgt` gives the density, `pgt` gives the distribution function, and `qgt` gives the quantile function

References

Y. Xiao (2018). On the Solution of a Generalized Behrens-Fisher Problem. *Far East Journal of Theoretical Statistics*, **54** (1), 21-140.

See Also

`dt`

Examples

```
## equal variances
dgt(0, n1 = 5, n2 = 10, v1tov2 = 1)
dt(0, df = 5+10-2)

pgt(2, n1 = 5, n2 = 10, v1tov2 = 1)
pt(2, df = 5+10-2)

qgt(0.975, n1 = 5, n2 = 10, v1tov2 = 1)
qt(0.975, df = 5+10-2)

library(ggplot2)
ggplot(data = data.frame(x = c(-4, 4)), aes(x)) +
    stat_function(fun = function(x) dgt(x, n1 = 5, n2 = 10, v1tov2 = 1),
                  n = 101, col = "darkblue", lwd = 1) +
    stat_function(fun = function(x) dt(x, df = 5+10-2),
                  n = 101, col = "darkgreen", lwd = 1) + ylab("density") +
    ggtitle("Generalized Central (blue) t vs. Student t Distribution (green)")


## unequal variances
nu <- (1^2/5+5^2/10)^2/(1^4/(5^2*(5-1)) + 5^4/(10^2*(10-1)))
dgt(0, n1 = 5, n2 = 10, v1tov2 = 1^2/5^2)
dt(0, df = nu)

pgt(2, n1 = 5, n2 = 10, v1tov2 = 1^2/5^2)
pt(2, df = nu)

qgt(0.975, n1 = 5, n2 = 10, v1tov2 = 1^2/5^2)
```

```
qt(0.975, df = nu)

ggplot(data = data.frame(x = c(-4, 4)), aes(x)) +
    stat_function(fun = function(x) dgt(x, n1 = 5, n2 = 10, v1tov2 = 1^2/5^2),
                  n = 101, col = "darkblue", lwd = 1) +
    stat_function(fun = function(x) dt(x, df = nu),
                  n = 101, col = "darkred", linetype = "dashed", lwd = 1) +
    ylab("density") +
    ggtitle("Generalized Central t (blue) vs. Welch t Distribution (red)")

## Not run:
  ## parallel computing
  pgt(seq(from = -2, to = 2, length = 30), n1 = 5, n2 = 10, v1tov2 = 1^2/5^2,
      parallel = TRUE)
  qgt(seq(from = 0.1, to = 0.9, length = 30), n1 = 5, n2 = 10, v1tov2 = 1^2/5^2,
+     parallel = TRUE)

## End(Not run)
```

`fingsys` *Systolic Blood Pressure Dataset used in Bland and Altman (1995)*

Description

This dataset was used in Band and Altman (1995) to demonstrate why plotting difference against standard method is misleading.

Usage

```
data(fingsys)
```

Format

A data.frame with two colums

`armsys` systolic blood pressure (mm Hg) measured by a standard arm cuff.

`fingsys` systolic blood pressure (mm Hg) measured by a finger monitor.

For more details see Close et al. (1986) as well as Bland and Altman (1995).

Details

The dataset is a random subset of 200 observations from a larger dataset of Close et al. (1986) that was used in Bland and Altman (1995) to demonstrate why plotting difference against standard method is misleading.

Source

The data set was obtained from https://www-users.york.ac.uk/~mb55/datasets/fingsys.dct

References

Close A, Hamilton G, Muriss S (1986). Finger systolic pressure: its use in screening for hypertension and monitoring. *Brit Med J* 293:775-778.

Bland JM, Altman DG. (1995) Comparing methods of measurement: why plotting difference against standard method is misleading. *Lancet* 346, 1085-7.

Examples

```
data(fingsys)
str(fingsys)
head(fingsys)
tail(fingsys)
```

`h0plot` *Visualize Distribution of Test Statistic.*

Description

The function visualizes the distribution of the test statistic.

Usage

```
h0plot(x, ...)

## Default S3 method:
h0plot(x, statistic, pval, alternative = "two.sided", sig.level = 0.05,
                    hist.alpha = 0.2, stat.col = "darkgreen", dens.col = "black",
                          dens.alpha = 0.5, cval.col = "darkred", lwd = 1,
                          breaks = NULL, digits = 3, ...)

## S3 method for class 'htest'
h0plot(x, sig.level = 0.05, hist.alpha = 0.2,
                         stat.col = "darkgreen", dens.col = "black",
                         dens.alpha = 0.5, cval.col = "darkred", lwd = 1,
                         breaks = NULL, digits = 3, qtail = 1e-3, n = 501, ...)

## S3 method for class 'perm.htest'
h0plot(x, sig.level = 0.05, hist.alpha = 0.2, stat.col = "darkgreen",
                              dens.col = "black", dens.alpha = 0.5,
                              cval.col = "darkred", lwd = 1, breaks = NULL,
                              digits = 3, ...)

## S3 method for class 'boot.htest'
h0plot(x, sig.level = 0.05, hist.alpha = 0.2, stat.col = "darkgreen",
                       dens.col = "black", dens.alpha = 0.5, cval.col = "darkred",
                              lwd = 1, breaks = NULL, digits = 3, ...)
```

Arguments

x	bootstrap/permuation results of the test statistic or result of class htest, perm.htest, or boot.htest.
statistic	test statistic of the test on the original data.
pval	p value of the 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.
sig.level	significance level of the test.
hist.alpha	opacity used for geom_histogram.
stat.col	color used for line representing the (final) test statistic.
dens.col	color used for the density line.
dens.alpha	opacity used for geom_area to color the rejection region.
cval.col	color used for rejection region (area under the density beyond the critical value).
lwd	line width used for line representing the (final) test statistic as well ase for geom_density.
breaks	argument breaks of geom_histogram.
digits	number of significant digits used for p values.
qtail	tail quantile to determine plot limits of x-axis.
n	integer; the number of x values at which to evaluate.
...	further arguments that may be passed through.

Details

The function can be used to plot the distribution of the test statistic under H0.

In case of class htest the implementation so far covers t.test, oneway.test, and var.test. In principle, the function should work for all tests of class htest returning a t or F statistic.

Value

ggplot object

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

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.

Examples

```
## Welch t-Test
h0plot(t.test(extra ~ group, data = sleep))
h0plot(t.test(extra ~ group, data = sleep, alternative = "less"))

## Hsu t-Test
h0plot(hsu.t.test(extra ~ group, data = sleep))
h0plot(hsu.t.test(extra ~ group, data = sleep, alternative = "less"))

## Welch 1-Way ANOVA
h0plot(oneway.test(extra ~ group, data = sleep))

## F test to compare two variances
h0plot(var.test(extra ~ group, data = sleep))

## Bootstrap based Welch t-Test
h0plot(boot.t.test(extra ~ group, data = sleep, bootStat = TRUE))
h0plot(boot.t.test(extra ~ group, data = sleep, bootStat = TRUE,
                   alternative = "less"))

## Permutation based Welch t-Test
h0plot(perm.t.test(extra ~ group, data = sleep, permStat = TRUE))
h0plot(perm.t.test(extra ~ group, data = sleep, permStat = TRUE,
                   alternative = "less"))
```

`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 NAs. 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.

Following the recommendation of Section 6.8.3 and 7.4.4.2 of Hedderich and Sachs (2018), one should at least have six observations per group to apply the test.

The test is conservative, but is a valid alternative when the sample sizes are very small in one group and larger in the second group as considered in Adusah and Brooks (2011).

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

A.K. Adusah, G.P. Brooks (2011). Type I Error Inflation of the Separate-Variances Welch t test with Very Small Sample Sizes when Assumptions Are Met. *Journal of Modern Applied Statistical Methods* **10**, 362-372.

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

P. Hsu (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(mtcars, t.test(mpg[am == 0], mpg[am == 1]))
with(mtcars, hsu.t.test(mpg[am == 0], mpg[am == 1]))
## Formula interface
t.test(mpg ~ am, data = mtcars)
hsu.t.test(mpg ~ am, data = mtcars)
```

---

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, corr)
```

Arguments

SD1	numeric vector, baseline SD.
SD2	numeric vector, follow-up SD.
SDchange	numeric vector, SD for changes from baseline.
corr	optional numeric vector of correlations; see details below.

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. Alternatively, correlations can be provided via argument corr. This option may particularly be useful, if no complete data is available.

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)
SDchange2 <- rep(NA, 9)
imputeSD(SD1, SD2, SDchange2, corr = c(0.85, 0.9, 0.95))
```

---

md2sens *Compute Sensitivity and Specificty based on MD.*

Description

The function calculates the sensitivity and specificity for a given mean difference (MD) and specified standard deviations of the two groups. This is achieved by using the optimal cut-off poin based on Youden's J statistic.

Usage

```
md2sens(delta, sd1 = 1, sd2 = 1)
```

Arguments

delta	(absolute) mean difference of the two groups.
sd1	standard deviation of first group.
sd2	standard deviation of second group.

Details

This function can be used to demonstrate how (standardized) effect sizes translate into sensitivity and specifity, based on two normal distributions.

The two standard diviations are assumed to be equal if their difference is smaller than 1e-08.

Value

Numeric vector with sensitivity and specificity.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

B\"ohning D, B\"ohning W, Holling H (2008). Revisiting Youden's index as a useful measure of the misclassification error in meta-analysis of diagnostic studies. Stat Methods Med Res. 17(6), 543-54.

J.H. Zhang, T.D.Y. Chung and K.R. Oldenburg (1999). A simple statistical parameter for use in evaluation and validation of high throughput screening assays. *Journal of Biomolecular Screening* **4** (2): 67-73.

Examples

```
## (standardized) mean difference to sensitivity/specificity
## equal variances
library(ggplot2)
delta <- seq(from = 0.0, to = 6, by = 0.05)
res <- sapply(delta, md2sens)
DF <- data.frame(SMD = delta, sensitivity = res[1,],
                 specificity = res[2,])
ggplot(DF, aes(x = SMD, y = sensitivity)) +
  geom_line() + ylim(0.5, 1.0) + xlab("(standardized) mean difference") +
  ylab("sensitivity = specificity") + ggtitle("SD1 = SD2 = 1")

## unequal variances
delta <- seq(from = 0.0, to = 6, by = 0.05)
res <- sapply(delta, md2sens, sd1 = 1, sd2 = 2)
DF <- data.frame(MD = delta, performance = c(res[1,], res[2,]),
                 measure = c(rep("sensitivity", length(delta)),
                             rep("specificity", length(delta))))
ggplot(DF, aes(x = MD, y = performance, color = measure)) +
  geom_line() + ylim(0, 1.0) + xlab("mean difference") +
```

```
  scale_color_manual(values = c("darkblue", "darkred")) +
  ggtitle("SD1 = 1, SD2 = 2")
```

`md2zfactor` *Compute z-Factor based on MD.*

Description

The function calculates the z-factor for a given mean difference (MD) and specified standard deviations of the two groups.

Usage

```
md2zfactor(delta, sd1 = 1, sd2 = 1)
```

Arguments

`delta`	(absolute) mean difference of the two groups.
`sd1`	standard deviation of first group.
`sd2`	standard deviation of second group.

Details

This function can be used to demonstrate how (standardized) effect sizes translate into z-factor, based on two normal distributions.

Value

Numeric vector with sensitivity and specificity.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

J.H. Zhang, T.D.Y. Chung and K.R. Oldenburg (1999). A simple statistical parameter for use in evaluation and validation of high throughput screening assays. *Journal of Biomolecular Screening* **4** (2): 67-73.

Examples

```
## (standardized) mean difference to sensitivity/specificity
## equal variances
library(ggplot2)
delta <- seq(from = 2, to = 18, by = 0.05)
res <- sapply(delta, md2zfactor)
DF <- data.frame(SMD = delta, zfactor = res)
ggplot(DF, aes(x = SMD, y = zfactor)) +
  geom_line() + xlab("(standardized) mean difference") +
  ylab("z-factor") + ggtitle("SD1 = SD2 = 1") +
  geom_hline(yintercept = 1, linetype = "dotted")

## unequal variances
delta <- seq(from = 2.5, to = 20, by = 0.05)
res <- sapply(delta, md2zfactor, sd1 = 1, sd2 = 2)
DF <- data.frame(MD = delta, zfactor = res)
ggplot(DF, aes(x = MD, y = zfactor)) +
  geom_line() + xlab("mean difference") +
  ylab("z-factor") + ggtitle("SD1 = 1, SD2 = 2") +
  geom_hline(yintercept = 1, linetype = "dotted")
```

---

mdplot *Visualize MD.*

Description

The function visualizes the mean difference (MD) of two groups, assuming two normal distributions. In addition, the sensitivity and specificity based on the optimal cutoff (Youden's J statistic) can be added.

Usage

```
mdplot(delta, sd1 = 1, sd2 = 1, constant = 3, n = 501, alpha = 0.25,
        lwd = 1, col1 = "darkblue", col2 = "darkred", xlim,
        addSens = TRUE, digits = 3)
```

Arguments

delta	(absolute) mean difference of the two groups.
sd1	standard deviation of first group.
sd2	standard deviation of second group.
constant	used to define the plot range (constant*sd).
n	integer; the number of values at which the two normal densities are evaluated.
alpha	opacity used for geom_area.
lwd	linewidth of the plotted lines.
col1	color of density of first group.

col2	color of density of second group.
xlim	limits of x-axis. If missing, calculated based on constant and the specified SDs.
addSens	logical; should sensitivity and specificity be added to the plot.
digits	number of digits used for sensitivity and specificity.

Details

The function can be used to demonstrate (standardized) effect sizes and how these effect sizes transform into sensitivities and specificities.

The two standard diviations are assumed to be equal if their difference is smaller than 1e-08.

Value

ggplot object

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

B\"ohning D, B\"ohning W, Holling H (2008). Revisiting Youden's index as a useful measure of the misclassification error in meta-analysis of diagnostic studies. Stat Methods Med Res. 17(6), 543-54.

J.H. Zhang, T.D.Y. Chung and K.R. Oldenburg (1999). A simple statistical parameter for use in evaluation and validation of high throughput screening assays. *Journal of Biomolecular Screening* **4** (2): 67-73.

Examples

```
## small effect size
mdplot(delta = 0.2)
## medium effect size
mdplot(delta = 0.5)
## large effect size
mdplot(delta = 0.8)
## z-factor = 0  (z-factor = 1 - 3*2*sd/delta); see Zhang et al. (1999)
mdplot(delta = 6)
## z-factor = 0.5  (z-factor = 1 - 3*2*sd/delta); see Zhang et al. (1999)
mdplot(delta = 12)

## unequal variances
mdplot(delta = 0.8, sd1 = 1, sd2 = 2)
mdplot(delta = 0.8, sd1 = 2, sd2 = 1)
```

`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, ...)

## S3 method for class 'amelia'
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, ...)

## S3 method for class 'mids'
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 and 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, response = 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(response ~ group, data = D)
## Intention to treat analysis (Multiple Imputation Welch two-sample t-test)
mi.t.test(res, x = "response", y = "group")

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

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

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

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

## Use mice to impute missing values
library(mice)
res.mice <- mice(D, m = 10, print = FALSE)
mi.t.test(res.mice, x = "response", y = "group")
```

`mi.wilcox.test` *Multiple Imputation Wilcoxon Rank Sum and Signed Rank Tests*

Description

Performs one and two sample Wilcoxon tests on multiple imputed datasets.

Usage

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

## Default S3 method:
mi.wilcox.test(miData, x, y = NULL,
        alternative = c("two.sided", "less", "greater"), mu = 0,
        paired = FALSE, exact = NULL, conf.int = TRUE,
        conf.level = 0.95, subset = NULL, ...)

## S3 method for class 'amelia'
mi.wilcox.test(miData, x, y = NULL,
        alternative = c("two.sided", "less", "greater"), mu = 0,
        paired = FALSE, exact = NULL, conf.int = TRUE,
        conf.level = 0.95, subset = NULL, ...)

## S3 method for class 'mids'
mi.wilcox.test(miData, x, y = NULL,
        alternative = c("two.sided", "less", "greater"), mu = 0,
        paired = FALSE, exact = NULL, conf.int = TRUE,
        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 specifying an optional location parameter.
`paired`	a logical indicating whether you want a paired test.
`exact`	a logical indicating whether an exact p-value should be computed.
`conf.int`	a logical indicating whether a confidence interval should be computed.
`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

For details about the tests see `wilcox.exact`

We use the median p rule (MPR) for the computation of the p value of the test; see Section 5.3.2 of van Buuren (2018) or Section 13.3 in Heymans and Eekhout (2019). The approach seems to

work well in many situations such as logistic regression (Eekhout et al. (2017)) or GAM (Bolt et al. (2022)). However, we are not aware of any work that has investigated the MPR approach for Wilcoxon tests. Hence, this function should be regarded as experimental.

We recommend to use an odd number of imputations.

Value

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

statistic	he value of the test statistic with a name describing it.
p.value	the p-value for the test.
pointprob	this gives the probability of observing the test statistic itself (called point-prob).
null.value	the location parameter mu.
alternative	a character string describing the alternative hypothesis.
method	a character string indicating what type of test was performed.
data.name	a character string giving the name(s) of the data.
conf.int	a confidence interval for the location parameter. (Only present if argument conf.int = TRUE.)
estimate	Hodges-Lehmann estimate of the location parameter. (Only present if argument conf.int = TRUE.)

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

van Buuren, S. (2018). *Flexible Imputation of Missing Data*. Chapman & Hall/CRC. https://stefvanbuuren.name/fimd/.

Heymans, M.W. and Eekhout, I. (2019). *Applied Missing Data Analysis With SPSS and (R)Studio*. Self-publishing. https://bookdown.org/mwheymans/bookmi/.

Eekhout, I, van de Wiel, MA, Heymans, MW (2017). Methods for significance testing of categorical covariates in logistic regression models after multiple imputation: power and applicability analysis. *BMC Med Res Methodol*, **17**, 1:129. doi:10.1186/s1287401704047

Bolt, MA, MaWhinney, S, Pattee, JW, Erlandson, KM, Badesch, DB, Peterson, RA (2022). Inference following multiple imputation for generalized additive models: an investigation of the median p-value rule with applications to the Pulmonary Hypertension Association Registry and Colorado COVID-19 hospitalization data. *BMC Med Res Methodol*, **22**, 1:148. doi:10.1186/s12874022-01613w.

See Also

wilcox.exact

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, response = c(x, y), pair = pair, group = g)

## Use Amelia to impute missing values
library(Amelia)
res <- amelia(D, m = 9, p2s = 0, idvars = "ID", noms = "group")

## Per protocol analysis (Exact Wilcoxon rank sum test)
library(exactRankTests)
wilcox.exact(response ~ group, data = D, conf.int = TRUE)
## Intention to treat analysis (Multiple Imputation Exact Wilcoxon rank sum test)
mi.wilcox.test(res, x = "response", y = "group")

## Specifying alternatives
mi.wilcox.test(res, x = "response", y = "group", alternative = "less")
mi.wilcox.test(res, x = "response", y = "group", alternative = "greater")

## One sample test
wilcox.exact(D$response[D$group == "yes"], conf.int = TRUE)
mi.wilcox.test(res, x = "response", subset = D$group == "yes")
mi.wilcox.test(res, x = "response", mu = -1, subset = D$group == "yes",
               alternative = "less")
mi.wilcox.test(res, x = "response", mu = -1, subset = D$group == "yes",
               alternative = "greater")

## paired test
wilcox.exact(D$response, D$pair, paired = TRUE, conf.int = TRUE)
mi.wilcox.test(res, x = "response", y = "pair", paired = TRUE)

## Use mice to impute missing values
library(mice)
res.mice <- mice(D, m = 9, print = FALSE)
mi.wilcox.test(res.mice, x = "response", y = "group")
```

`mpe.t.test` *Intersection-Union t-Test for Testing Multiple Co-Primary Endpoints*

Description

The function computes the intersection-union t-test.

Usage

```
mpe.t.test(X, Y, conf.level = 0.975)
```

Arguments

X	matrix with observations of group 1 in rows
Y	matrix with obersvations of group 2 in rows
conf.level	confidence level of the interval.

Details

The function computes the intersection-union t-test. The implementation is based on the formulas given in the references below.

The null hypothesis reads $\mu_{Tk} - \mu_{Ck} \leq 0$ for at least one $k \in \{1, \ldots, K\}$ where Tk is treatment k, Ck is control k and K is the number of co-primary endpointss (i.e. number of columns of X and Y).

Value

Object of class "mpe.test".

Note

The function first appeared in package **mpe**, which is now archived on CRAN.

Author(s)

Srinath Kolampally, Matthias Kohl <Matthias.Kohl@stamats.de>

References

Sugimoto, T. and Sozu, T. and Hamasaki, T. (2012). A convenient formula for sample size calculations in clinical trials with multiple co-primary continuous endpoints. *Pharmaceut. Statist.*, **11**: 118-128. doi:10.1002/pst.505

Sozu, T. and Sugimoto, T. and Hamasaki, T. and Evans, S.R. (2015). *Sample Size Determination in Clinical Trials with Multiple Endpoints*. Springer Briefs in Statistics, ISBN 978-3-319-22005-5.

See Also

mpe.z.test

Examples

```
library(mvtnorm)
delta <- c(0.25, 0.5)
Sigma <- matrix(c(1, 0.75, 0.75, 1), ncol = 2)
n <- 50
X <- rmvnorm(n=n, mean = delta, sigma = Sigma)
Y <- rmvnorm(n=n, mean = rep(0, length(delta)), sigma = Sigma)
mpe.t.test(X = X, Y = Y)
```

`mpe.z.test` *Intersection-Union z-Test for Testing Multiple Co-Primary Endpoints*

Description

The function computes the intersection-union z-test.

Usage

```
mpe.z.test(X, Y, Sigma, conf.level = 0.975)
```

Arguments

`X`	matrix with observations of group 1 in rows
`Y`	matrix with obersvations of group 2 in rows
`Sigma`	known covariance matrix.
`conf.level`	confidence level of the interval.

Details

The function computes the intersection-union z-test. The implementation is based on the formulas given in the references below.

The null hypothesis reads $\mu_{Tk} - \mu_{Ck} \leq 0$ for at least one $k \in \{1, \ldots, K\}$ where Tk is treatment k, Ck is control k and K is the number of co-primary endpoints (i.e. number of columns of `X` and `Y`).

Value

Object of class `"mpe.test"`.

Note

The function first appeared in package **mpe**, which is now archived on CRAN.

Author(s)

Srinath Kolampally, Matthias Kohl <Matthias.Kohl@stamats.de>

References

Sugimoto, T. and Sozu, T. and Hamasaki, T. (2012). A convenient formula for sample size calculations in clinical trials with multiple co-primary continuous endpoints. *Pharmaceut. Statist.*, **11**: 118-128. doi:10.1002/pst.505

Sozu, T. and Sugimoto, T. and Hamasaki, T. and Evans, S.R. (2015). *Sample Size Determination in Clinical Trials with Multiple Endpoints*. Springer Briefs in Statistics, ISBN 978-3-319-22005-5.

See Also

mpe.t.test

Examples

```
library(mvtnorm)
delta <- c(0.25, 0.5)
Sigma <- matrix(c(1, 0.75, 0.75, 1), ncol = 2)
n <- 50
X <- rmvnorm(n=n, mean = delta, sigma = Sigma)
Y <- rmvnorm(n=n, mean = rep(0, length(delta)), sigma = Sigma)
mpe.z.test(X = X, Y = Y, Sigma = Sigma)
```

---

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.
...	further arguments passed to function boot, e.g. for parallel computing.

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)


## parallel computing for bootstrap
normCI(x, boot = TRUE, R = 9999, parallel = "multicore", ncpus = 2)
```

---

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.
...	further arguments passed to function boot, e.g. for parallel computing.

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"`, `"hsu"` and `"xiao"` allow for unequal variances. `"welch"` and `"hsu"` 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). Method `"xiao"` is based on the quantiles of the generalized t-distribution introduced in Section 5.2 of Xiao (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.

Y. Xiao (2018). On the Solution of a Generalized Behrens-Fisher Problem. *Far East Journal of Theoretical Statistics*, **54** (1), 21-140.

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")
## Xiao
normDiffCI(x, y, method = "xiao")
## In case of bootstrap there is no difference between welch, hsu and xiao
## (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")


## parallel computing for bootstrap
normDiffCI(x, y, method = "welch", boot = TRUE, R = 9999,
           parallel = "multicore", ncpus = 2)



## Monte-Carlo simulation: coverage probability
M <- 100 # increase for more stable/realistic results!
CIxiao <- 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")$conf.int
  CIxiao[i,] <- normDiffCI(x, y, method = "xiao")$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
## Xiao
sum(CIxiao[,1] < 0 & 0 < CIxiao[,2])/M
```

`p2ses` *Compute SES from p value.*

Description

The function computes the SES (standardized effect size) from the p value for permutation/randomisation tests as proposed by Botta-Dukat (2018).

Usage

```
p2ses(p, alternative = c("two.sided", "less", "greater"))
```

Arguments

p	numeric vector of p 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.

Details

The function uses the probit transformation (qnorm) to compute an alternative SES based on p values from a permutation/randomization test as proposed by Botta-Dukat (2018) for skewed distributions.

Value

SES

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Botta-Dukat, Z (2018). Cautionary note on calculating standardized effect size (SES) in randomization test. Community Ecology 19(1), 77-83.

Examples

```
## symmetric case
x <- rnorm(100)
res <- perm.t.test(x)
p2ses(res$p.value)
abs(res$statistic)

## skewed case
x <- rgamma(100, shape = 5)
res <- perm.t.test(x, mu = 5)
p2ses(res$p.value)
abs(res$statistic)
```

`pairwise.ext.t.test` *Compute Pairwise t Tests*

Description

The function computes pairwise t tests using functions `t.test`, `boot.t.test` or `perm.t.test`.

Usage

```
pairwise.ext.t.test(x, g, method = "t.test", p.adjust.method = "holm",
                    paired = FALSE, ...)
```

Arguments

`x`	numeric vector.
`g`	grouping vector or factor
`method`	character giving the name of the function to be applied; that is, `"t.test"`, `"hsu.t.test"`, `"xiao.t.test"`, `"boot.t.test"` or `"perm.t.test"`.
`p.adjust.method`	method for adjusting p values (see `p.adjust`). Can be abbreviated.
`paired`	a logical indicating whether you want a paired test.
`...`	additional arguments to fun.

Details

The function computes pairwise t tests using function `t.test`, `hsu.t.test`, `xiao.t.test`, `boot.t.test` or `perm.t.test`.

The implementation is in certain aspects analogously to `pairwise.t.test`. However, a more detailed output is generated.

Value

Object of class `"pw.htest"` containing the following components:

`data.name`	a character string giving the names of the data.
`method`	the type of test applied.
`null.value`	the location parameter mu.
`alternative`	a character string describing the alternative hypothesis.
`conf.level`	confidence level of the confidence interval.
`results`	a data.frame containing the results of function `t.test`, `boot.t.test` or `perm.t.test`.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

t.test, 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.ext.t.test(x, g)
## in contrast to
pairwise.t.test(x, g, pool.sd = FALSE)
## moreover
pairwise.ext.t.test(x, g, method = "hsu.t.test")
pairwise.ext.t.test(x, g, method = "xiao.t.test")
pairwise.ext.t.test(x, g, method = "boot.t.test")
pairwise.ext.t.test(x, g, method = "perm.t.test")
```

---

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)
```

---

pairwise.wilcox.exact *Compute Pairwise Wilcoxon Tests*

Description

The function computes pairwise Wilcoxon rank sum and signed rank tests using function wilcox.exact of package exactRankTests.

Usage

```
pairwise.wilcox.exact(x, g, p.adjust.method = "holm", paired = FALSE, ...)
```

Arguments

x	numeric vector.
g	grouping vector or factor
p.adjust.method	method for adjusting p values (see p.adjust). Can be abbreviated.
paired	a logical indicating whether you want a paired test.
...	additional arguments to fun.

Details

The function computes pairwise Wilcoxon rank sum and signed rank tests.

The implementation is in certain aspects analogously to pairwise.wilcox.test. However, a more detailed output is generated.

Value

Object of class "pw.htest" containing the following components:

data.name	a character string giving the names of the data.
method	the type of test applied.
null.value	the location parameter mu.
alternative	a character string describing the alternative hypothesis.
conf.level	confidence level of the confidence interval.
results	a data.frame containing the results of function wilcox.exact

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

wilcox.exact, pairwise.wilcox.test

Examples

```
set.seed(13)
x <- rnorm(100)
g <- factor(sample(1:4, 100, replace = TRUE))
levels(g) <- c("a", "b", "c", "d")
pairwise.wilcox.exact(x, g)
## in contrast to
pairwise.wilcox.test(x, g)
```

---

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.type = "pivot", conf.level = 0.95, R = 9999, symmetric = TRUE,
       permStat = FALSE, useCombn = FALSE, ...)

## 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.type	a character string specifying which type of confidence interval to compute; see details below.
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 NAs. Defaults to getOption("na.action").
permStat	logical; should the permutation results for the test statistic be returned. If TRUE, function h0plot can be used to plot the distribution of the results.
useCombn	logical; use all combinations instead of permutations in the case of the two-sample t-test; see Details.
...	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.

Janssen and Pauls (2003) state in Example 5 (c) about Bootstrap two-sample tests: "From the previous comments it is clear that permutation tests should always be preferred for two-sample problems."

The function returns permutation p values and confidence intervals as well as the results ot the t-test without permutations. If useCombn is TRUE in the two-sample case, then all combinations instead of all permutations are used. This option only has an effect for small sample sizes when the number of

all combinations/permutations is smaller than R. In principle, the results should be identical in this case. However, the difference in the number of permutations/combinations can have a minor effect on p values, confidence intervals as well as the distribution of the test statistic.

The function computes four types of permutation confidence intervals:

pivot Pivot-Inverted Test Confidence Interval (Garthwaite (1996), Good (2005)). Constructed by inverting the studentized permutation test while treating the sample permutation distribution as fixed (pivotal). It provides a fast, stable, and smooth approximation of test inversion.

exact Exact Inverted Test Confidence Interval (Chung and Romano (2013)). Obtained by exact test inversion where the permutation standard error is dynamically re-computed under each null hypothesis shift. It guarantees exact coverage under the i.i.d. assumption and asymptotic validity under heteroscedasticity, though extreme outliers may result in conservative (wide) bounds.

stud Studentized (Bootstrap-t) Permutation Confidence Interval (Hall (1988), Janssen (1997)). Uses the quantiles of the studentized permutation distribution centered at 0 and scales them by the observed standard error.

perc Percentile Permutation Confidence Interval (Efron (1981), Efron and Tibshirani (1993)). Derived directly from the empirical quantiles of the unstudentized permutation distribution of the effect size.

all All types of confidence intervals are computed.

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.
perm.estimate	(Monte-Carlo) permutation estimate.

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.
perm.stderr	(Monte-Carlo) permutation standard error.
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.
perm.statistic	NULL or numeric vector with permutation results.

Note

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

References

B. Efron (1981). Nonparametric standard errors and confidence intervals. *The Canadian Journal of Statistics*, **9**(2), 139-158.

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

P.H. Garthwaite (1996). Confidence intervals from randomization tests. *Biometrics*, **52**(4), 1387-1393.

P. Good (2005). *Permutation, Parametric, and Bootstrap Tests of Hypotheses* (3rd ed.). Springer.

P. Hall (1986). On the bootstrap and confidence intervals. *The Annals of Statistics*, **14**(4), 1431-1452.

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.

A. Janssen, T. Pauls. How do bootstrap and permutation tests work?. *Ann. Statist.*, **31**(3), 768-806.

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), conf.type = "all")

t.test(1:10, y = c(7:20, 200)) # P = .1245    -- NOT significant anymore
perm.t.test(1:10, y = c(7:20, 200), conf.type = "pivot")
perm.t.test(1:10, y = c(7:20, 200), conf.type = "exact")
perm.t.test(1:10, y = c(7:20, 200), conf.type = "stud")
perm.t.test(1:10, y = c(7:20, 200), conf.type = "perc") # p < 1e-04, but CI covers 0
```

```
## Traditional interface
with(mtcars, t.test(mpg[am == 0], mpg[am == 1]))
with(mtcars, perm.t.test(mpg[am == 0], mpg[am == 1]))
## Formula interface
t.test(mpg ~ am, data = mtcars)
perm.t.test(mpg ~ am, data = mtcars)


## all permutations vs combinations
perm.t.test(1:3, y = 2:5, useCombn = TRUE)
perm.t.test(1:3, y = 2:5, useCombn = FALSE)
```

`print.mpe.test` *Print Method for MPE Hypothesis Test*

Description

Printing objects of class "mpe.test" by simple print methods.

Usage

```
  ## S3 method for class 'mpe.test'
print(x, digits = getOption("digits"), prefix = "\t", ...)
```

Arguments

x	object of class "mpe.test".
digits	number of significant digits to be used.
prefix	string, passed to strwrap for displaying the method component of the mpe.test object.
...	further arguments to be passed to or from methods.

Details

The print method is based on the respective method print.htest of package **stats**.

Value

the argument x, invisibly, as for all print methods.

Note

The function first appeared in package **mpe**, which is now archived on CRAN.

Author(s)

Srinath Kolampally, Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

print.power.htest, mpe.z.test, mpe.t.test.

Examples

```
library(mvtnorm)
delta <- c(0.25, 0.5)
Sigma <- matrix(c(1, 0.75, 0.75, 1), ncol = 2)
n <- 50
X <- rmvnorm(n=n, mean = delta, sigma = Sigma)
Y <- rmvnorm(n=n, mean = rep(0, length(delta)), sigma = Sigma)
mpe.t.test(X = X, Y = Y)
```

---

quantileCI *Confidence Intervals for Quantiles*

Description

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

Usage

```
quantileCI(x, prob = 0.5, conf.level = 0.95, method = "exact",
           R = 9999, bootci.type = c("norm", "basic", "perc", "bca"),
           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"),
         na.rm = FALSE, alternative = c("two.sided", "less", "greater"), ...)
madCI(x, conf.level = 0.95, method = "exact",
      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.
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.
...	further arguments passed to function boot, e.g. for parallel computing.

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, the shortest confidence interval 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).

Since madCI is computed as the median confidence interval of the absolut deviations from the sample median and ignores the variablity of the sample median, the exact and asymptotic confidence intervals might be too short. We recommend to use bootstrap confidence intervals.

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

quantile

Examples

```
## To get a non-trivial exact confidence interval for the median
## one needs at least 6 observations
x <- rnorm(5)
medianCI(x)

## 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)


## parallel computing for bootstrap
medianCI(x, method = "boot", R = 9999, parallel = "multicore",
         ncpus = 2)
```

`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 and 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 and 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 and 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,
                              na.rm = TRUE, ...)
```

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	blending factor (default: no blending.
shape	point shape used.
na.rm	single logical, remove NA values before plotting.
...	further arguments that may be passed through.

Details

The plot generates a ggplot2 object that is shown.

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")
```

---

xiao.t.test *Xiao Two-Sample t-Test*

Description

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

Usage

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

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

## S3 method for class 'formula'
xiao.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 NAs. Defaults to getOption("na.action").
...	further arguments to be passed to or from methods.

Details

The function and its documentation was adapted from hsu.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.

The test is based on the solution of a generalized Behrens-Fisher problem derived by Xiao (2018) and corresponds to the two-sample test derived in Section 5.3 of Xiao (2018). The test is based on a generalized t-distribution.

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

Y. Xiao (2018). On the Solution of a Generalized Behrens-Fisher Problem. *Far East Journal of Theoretical Statistics*, **54** (1), 21-140.

See Also

t.test, hsu.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
xiao.t.test(1:10, y = c(7:20))
xiao.t.test(1:10, y = c(7:20, 200))

## Traditional interface
with(mtcars, t.test(mpg[am == 0], mpg[am == 1]))
with(mtcars, xiao.t.test(mpg[am == 0], mpg[am == 1]))
## Formula interface
t.test(mpg ~ am, data = mtcars)
xiao.t.test(mpg ~ am, data = mtcars)

## Example 6.1 in Xiao (2018)
x <- c(134, 146, 104, 119, 124, 161, 107, 83, 113, 129, 97, 123)
y <- c(70, 118, 101, 85, 107, 132, 94)
xiao.t.test(x, y, alternative = "greater")
t.test(x, y, var.equal = TRUE, alternative = "greater")
t.test(x, y, alternative = "greater")
hsu.t.test(x, y, alternative = "greater")
```

Index

∗ **datasets**
fingsys, 17
∗ **distribution**
dgt, 15
∗ **hplot**
baplot, 3
h0plot, 18
mdplot, 26
volcano, 54
∗ **htest**
boot.t.test, 10
hsu.t.test, 20
mi.t.test, 28
mi.wilcox.test, 30
mpe.t.test, 33
mpe.z.test, 35
perm.t.test, 45
print.mpe.test, 49
rm.oneway.test, 52
xiao.t.test, 55
∗ **multivariate**
mpe.t.test, 33
mpe.z.test, 35
∗ **package**
MKinfer-package, 2
∗ **power.htest**
print.mpe.test, 49
∗ **univar**
binomCI, 5
binomDiffCI, 8
cvCI, 13
imputeSD, 22
md2sens, 23
md2zfactor, 25
normCI, 36
normDiffCI, 38
p2ses, 40
pairwise.ext.t.test, 42
pairwise.fun, 43
pairwise.wilcox.exact, 44
quantileCI, 50

aov, *53*

baplot, 3
binom.test, *7*
binomCI, 5
binomDiffCI, 8
boot, *4*, *6*, *8*, *14*, *36*, *38*, *51*
boot.ci, *4*, *6*, *8*, *9*, *14*, *36–38*, *50*
boot.t.test, 10, *42*, *48*

CV, *14*
cvCI, 13

dgt, 15
dt, *16*

fingsys, 17
friedman.test, *53*

h0plot, 18
hsu.t.test, 20, *42*, *56*, *57*
hypergeo, *15*

imputeSD, 22
integrate, *15*, *16*

lme, *53*

mad, *50*
madCI *(*quantileCI*)*, 50
makeCluster, *15*
md2sens, 23
md2zfactor, 25
mdplot, 26
meanCI, *13*, *48*
meanCI *(*normCI*)*, 36
meanDiffCI, *13*, *48*
meanDiffCI *(*normDiffCI*)*, 38

medianCI *(quantileCI)*, 50
mi.t.test, 28
mi.wilcox.test, 30
MKinfer *(MKinfer-package)*, 2
MKinfer-package, 2
model.frame, *11*, *21*, *46*, *56*
mpe.t.test, 33, *36*, *50*
mpe.z.test, *34*, 35, *50*

normCI, 36
normDiffCI, 38

p.adjust, *42*, *44*
p2ses, 40
pairwise.ext.t.test, 42
pairwise.fun, 43
pairwise.t.test, *42–44*
pairwise.wilcox.exact, 44
pairwise.wilcox.test, *44, 45*
parallel, *15*
parRapply, *15*
perm.t.test, *13*, *42*, 45
pgt *(dgt)*, 15
print, *49*
print.mpe.test, 49
print.power.htest, *50*
prop.test, *9*

qgt *(dgt)*, 15
quade.test, *53*
quantile, *51*
quantileCI, 50

rm.oneway.test, 52

sdCI *(normCI)*, 36
snow, *15*
strwrap, *49*

t.test, *12, 13*, *21, 22*, *29*, *42, 43*, *48*, *57*

uniroot, *15, 16*

volcano, 54

wilcox.exact, *31, 32*, *45*

xiao.t.test, *42*, 55