Package ‘nptest’

March 1, 2021

Type Package

Title Nonparametric Bootstrap and Permutation Tests

Version 1.0-3

Date 2021-03-01

Author Nathaniel E. Helwig <helwig@umn.edu>

Maintainer Nathaniel E. Helwig <helwig@umn.edu>

Depends parallel

Description Robust nonparametric bootstrap and permutation tests for location, correlation, and regression problems, as described in Helwig (2019a) <doi:10.1002/wics.1457> and Helwig (2019b) <doi:10.1016/j.neuroimage.2019.116030>. Univariate and multivariate tests are supported. For each problem, exact tests and Monte Carlo approximations are available. Five different nonparametric bootstrap confidence intervals are implemented. Parallel computing is implemented via the 'parallel' package.

NeedsCompilation no

Repository CRAN

Date/Publication 2021-03-01 06:40:02 UTC

R topics documented:

`nptest-package` *Nonparametric Bootstrap and Permutation Tests*

Description

Robust nonparametric bootstrap and permutation tests for location, correlation, and regression problems, as described in Helwig (2019a) <doi:10.1002/wics.1457> and Helwig (2019b) <doi:10.1016/j.neuroimage.2019.116030 Univariate and multivariate tests are supported. For each problem, exact tests and Monte Carlo approximations are available. Five different nonparametric bootstrap confidence intervals are implemented. Parallel computing is implemented via the 'parallel' package.

Details

The DESCRIPTION file:

Package:	nptest
Type:	Package
Title:	Nonparametric Bootstrap and Permutation Tests
Version:	1.0-3
Date:	2021-03-01
Author:	Nathaniel E. Helwig <helwig@umn.edu>
Maintainer:	Nathaniel E. Helwig <helwig@umn.edu>
Depends:	parallel
Description:	Robust nonparametric bootstrap and permutation tests for location, correlation, and regression problems, as des
License:	GPL (>=2)

Index of help topics:

```
flipn                   Generate All Sign-Flips of n Elements
mcse                    Monte Carlo Standard Errors for Tests
np.boot                 Nonparametric Bootstrap Resampling
np.cor.test             Nonparametric Tests of Correlation Coefficients
np.loc.test             Nonparametric Tests of Location Parameters
np.reg.test             Nonparametric Tests of Regression Coefficients
nptest-package          Nonparametric Bootstrap and Permutation Tests
permn                   Generate All Permutations of n Elements
plot                    Plots Permutation Distribution for
                        Nonparametric Tests
```

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Maintainer: Nathaniel E. Helwig <helwig@umn.edu>

References

Blair, R. C., Higgins, J. J., Karniski, W., & Kromrey, J. D. (1994). A study of multivariate permutation tests which may replace Hotelling's T2 test in prescribed circumstances. Multivariate Behavioral Research, 29(2), 141-163. doi: 10.1207/s15327906mbr2902_2

Carpenter, J., & Bithell, J. (2000). Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Statistics in Medicine, 19*(9), 1141-1164. doi: 10.1002/(SICI)1097-0258(20000515)19:9%3C1141::AID-SIM479%3E3.0.CO;2-F

Chung, E., & Romano, J. P. (2016). Asymptotically valid and exact permutation tests based on two-sample U-statistics. Journal of Statistical Planning and Inference, 168, 97-105. doi: 10.1016/j.jspi.2015.07.004

Davison, A. C., & Hinkley, D. V. (1997). *Bootstrap Methods and Their Application*. Cambridge University Press. doi: 10.1017/CBO9780511802843

DiCiccio, C. J., & Romano, J. P. (2017). Robust permutation tests for correlation and regression coefficients. Journal of the American Statistical Association, 112(519), 1211-1220. doi: 10.1080/01621459.2016.1202117

Draper, N. R., & Stoneman, D. M. (1966). Testing for the inclusion of variables in linear regression by a randomisation technique. Technometrics, 8(4), 695-699. doi: 10.2307/1266641

Efron, B., & Tibshirani, R. J. (1994). *An Introduction to the Boostrap*. Chapman & Hall/CRC. doi: 10.1201/9780429246593

Fisher, R. A. (1925). Statistical methods for research workers. Edinburgh: Oliver and Boyd.

Freedman, D., & Lane, D. (1983). A nonstochastic interpretation of reported significance levels. Journal of Business and Economic Statistics, 1(4), 292-298. doi: 10.2307/1391660

Helwig, N. E. (2019a). Statistical nonparametric mapping: Multivariate permutation tests for location, correlation, and regression problems in neuroimaging. WIREs Computational Statistics, 11(2), e1457. doi: 10.1002/wics.1457

Helwig, N. E. (2019b). Robust nonparametric tests of general linear model coefficients: A comparison of permutation methods and test statistics. NeuroImage, 201, 116030. doi: 10.1016/j.neuroimage.2019.116030

Huh, M.-H., & Jhun, M. (2001). Random permutation testing in multiple linear regression. Communications in Statistics - Theory and Methods, 30(10), 2023-2032. doi: 10.1081/STA-100106060

Janssen, A. (1997). Studentized permutation tests for non-i.i.d. hypotheses and the generalized Behrens-Fisher problem. Statistics & Probability Letters , 36 (1), 9-21. doi: 10.1016/S0167-7152(97)00043-6

Johnson, N. J. (1978). Modified t tests and confidence intervals for asymmetrical populations. Journal of the American Statistical Association, 73 (363), 536-544. doi: 10.2307/2286597

Kennedy, P. E., & Cade, B. S. (1996). Randomization tests for multiple regression. Communications in Statistics - Simulation and Computation, 25(4), 923-936. doi: 10.1080/03610919608813350

Manly, B. (1986). Randomization and regression methods for testing for associations with geographical, environmental and biological distances between populations. Researches on Population Ecology, 28(2), 201-218. doi: 10.1007/BF02515450

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. Annals Of Mathematical Statistics, 18(1), 50-60. doi: 10.1214/aoms/1177730491

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2018). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-0. https://CRAN.R-project.org/package=e1071

Nichols, T. E., Ridgway, G. R., Webster, M. G., & Smith, S. M. (2008). GLM permutation: non-parametric inference for arbitrary general linear models. NeuroImage, 41(S1), S72.

O'Gorman, T. W. (2005). The performance of randomization tests that use permutations of independent variables. Communications in Statistics - Simulation and Computation, 34(4), 895-908. doi: 10.1080/03610910500308230

Pitman, E. J. G. (1937a). Significance tests which may be applied to samples from any populations. Supplement to the Journal of the Royal Statistical Society, 4(1), 119-130. doi: 10.2307/2984124

Pitman, E. J. G. (1937b). Significance tests which may be applied to samples from any populations. ii. the correlation coefficient test. Supplement to the Journal of the Royal Statistical Society, 4(2), 225-232. doi: 10.2307/2983647

Romano, J. P. (1990). On the behavior of randomization tests without a group invariance assumption. Journal of the American Statistical Association, 85(411), 686-692. doi: 10.1080/01621459.1990.10474928

Still, A. W., & White, A. P. (1981). The approximate randomization test as an alternative to the F test in analysis of variance. British Journal of Mathematical and Statistical Psychology, 34(2), 243-252. doi: 10.1111/j.2044-8317.1981.tb00634.x

Student. (1908). The probable error of a mean. Biometrika, 6(1), 1-25. doi: 10.2307/2331554

ter Braak, C. J. F. (1992). Permutation versus bootstrap significance tests in multiple regression and ANOVA. In K. H. J\"ockel, G. Rothe, & W. Sendler (Eds.), Bootstrapping and related techniques. lecture notes in economics and mathematical systems, vol 376 (pp. 79-86). Springer.

Welch, B. L. (1938). The significance of the difference between two means when the population variances are unequal. Biometrika, 39(3/4), 350-362. doi: 10.2307/2332010

Wilcoxon, F. (1945). Individual comparisons by ranking methods. Biometrics Bulletin, 1(6), 80-83. doi: 10.2307/3001968

White, H. (1980). A heteroscedasticity-consistent covariance matrix and a direct test for heteroscedasticity. Econometrica, 48(4), 817-838. doi: 10.2307/1912934

Winkler, A. M., Ridgway, G. R., Webster, M. A., Smith, S. M., & Nichols, T. E. (2014). Permutation inference for the general linear model. NeuroImage, 92, 381-397. doi: 10.1016/j.neuroimage.2014.01.060

Examples

```
# See examples for...
#   flipn        (generate all sign flip vectors)
#   mcse         (Monte Carlo standard errors)
#   np.boot      (nonparametric bootstrap resampling)
#   np.cor.test  (nonparametric correlation tests)
#   np.loc.test  (nonparametric location tests)
#   np.reg.test  (nonparametric regression tests)
#   permn        (generate all permutation vectors)
```

`flipn` *Generate All Sign-Flips of n Elements*

Description

Generates all 2^n vectors of length n consisting of the elements -1 and 1.

Usage

```
flipn(n)
```

Arguments

`n`	Number of elements.

Details

Adapted from the "bincombinations" function in the e1071 R package.

Value

Matrix of dimension n by 2^n where each column contains a unique sign-flip vector.

Warning

For large `n` this function will consume a lot of memory and may even crash R.

Note

Used for exact tests in `np.loc.test` and `np.reg.test`.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2018). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-0. https://CRAN.R-project.org/package=e1071

Examples

```
flipn(2)
flipn(3)
```

mcse — *Monte Carlo Standard Errors for Tests*

Description

This function calculates Monte Carlo standard errors for (non-exact) nonparametric tests. The MCSEs can be used to determine (i) the accuracy of a test for a given number of resamples, or (ii) the number of resamples needed to achieve a test with a given accuracy.

Usage

```
mcse(R, delta, conf.level = 0.95, sig.level = 0.05,
     alternative = c("two.sided", "one.sided"))
```

Arguments

`R`	Number of resamples (positive integer).
`delta`	Accuracy of the approximation (number between 0 and 1).
`conf.level`	Confidence level for the approximation (number between 0 and 1).
`sig.level`	Significance level of the test (number between 0 and 1).
`alternative`	Alternative hypothesis (two-sided or one-sided).

Details

Note: either `R` or `delta` must be provided.

Let $F(x)$ denote the distribution function for the full permutation distribution, and let $G(x)$ denote the approximation obtained from R resamples. The *Monte Carlo standard error* is given by

$$\sigma(x) = \sqrt{F(x)[1 - F(x)]/R}$$

which is the standard deviation of $G(x)$.

A symmetric confidence interval for $F(x)$ can be approximated as

$$G(x) + / - C\sigma(x)$$

where C is some quantile of the standard normal distribution. Note that the critical value C corresponds to the confidence level (`conf.level`) of the approximation.

Let α denote the significance level (`sig.level`) for a one-sided test (α is one-half the significance level for two-sided tests). Define a to be the value of the test statistic such that $F(a) = \alpha$.

The parameter δ (`delta`) quantifies the accuracy of the approximation, such that

$$|G(a) - \alpha| < \alpha\delta$$

with a given confidence, which is controlled by the `conf.level` argument.

Value

`mcse`	Monte Carlo standard error.
`R`	Number of resamples.
`delta`	Accuracy of approximation.
`conf.level`	Confidence level.
`sig.level`	Significance level.
`alternative`	Alternative hypothesis.

Note

This function is only relevant for non-exact tests. For exact tests, $F(x) = G(x)$ so the Monte Carlo standard error is zero.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E. (2019). Statistical nonparametric mapping: Multivariate permutation tests for location, correlation, and regression problems in neuroimaging. WIREs Computational Statistics, 11(2), e1457. doi: 10.1002/wics.1457

See Also

`np.cor.test`, `np.loc.test`, `np.reg.test`

Examples

```
######***######   EXAMPLE 1   ######***######

# get the Monte Carlo standard error and the
# accuracy (i.e., delta) for given R = 10000
# using the default two-sided alternative hypothesis,
# the default confidence level (conf.level = 0.95),
# and the default significance level (sig.level = 0.05)

mcse(R = 10000)

# se = 0.0016
# delta = 0.1224


######***######   EXAMPLE 2   ######***######

# get the Monte Carlo standard error and the
# number of resamples (i.e., R) for given delta = 0.01
# using a one-sided alternative hypothesis,
```

```
# the default confidence level (conf.level = 0.95),
# and the default significance level (sig.level = 0.05)

mcse(delta = 0.1, alternative = "one.sided")

# se = 0.0026
# R = 7299
```

---

np.boot *Nonparametric Bootstrap Resampling*

Description

Nonparametric bootstrap resampling for univariate and multivariate statistics. Computes bootstrap estimates of the standard error, bias, and covariance. Also computes five different types of bootstrap confidence intervals: normal approximation interval, basic (reverse percentile) interval, percentile interval, studentized (bootstrap-*t*) interval, and bias-corrected and accelerated (BCa) interval.

Usage

```
np.boot(x, statistic, ..., R = 9999, level = c(0.9, 0.95, 0.99),
        method = c("norm", "basic", "perc", "stud", "bca")[-4],
        sdfun = NULL, sdrep = 99, jackknife = NULL,
        parallel = FALSE, cl = NULL, boot.dist = TRUE)
```

Arguments

x	vector of data (for univariate data) or vector of row indices (for multivariate data). See examples for bootstrapping multivariate data.
statistic	function that takes in x (and possibly additional arguments passed using ...) and returns a vector containing the statistic(s). See examples.
...	additional named arguments for the statistic function.
R	number of bootstrap replicates
level	desired confidence level(s) for the computed intervals. Default computes 90%, 95%, and 99% confidence intervals.
method	method(s) for computing confidence intervals. Partial matching is allowed. Any subset of allowable methods is permitted (default computes all intervals except studentized). Set method = NULL to produce no confidence intervals.
sdfun	function for computing the standard deviation of statistic. Should produce a vector the same length as the output of statistic. Only applicable if "stud" %in% method. If NULL, an inner bootstrap is used to estimate the standard deviation.
sdrep	number of bootstrap replicates for the inner bootstrap used to estimate the standard deviation of statistic. Only applicable if "stud" %in% method and sdfun = NULL. Larger values produce more accurate estimates (see Note).

`jackknife`	function that takes in `x` (and possibly additional arguments passed using `...`) and returns a vector containing the jackknife statistic(s). Should produce a vector the same length as the output of `statistic`. Only applicable if `"bca" %in% method`. If `NULL`, the jackknife function is defined as the statistic function (default). See the last example for a case when `statistic` and `jackknife` are different.
`parallel`	Logical indicating if the `parallel` package should be used for parallel computing (of the bootstrap distribution). Defaults to `FALSE`, which implements sequential computing.
`cl`	Cluster for parallel computing, which is used when `parallel = TRUE`. Note that if `parallel = TRUE` and `cl = NULL`, then the cluster is defined as `makeCluster(detectCores())`.
`boot.dist`	Logical indicating if the bootstrap distribution should be returned (see Note).

Details

The first three intervals (normal, basic, and percentile) are only first-order accurate, whereas the last two intervals (studentized and BCa) are both second-order accurate. Thus, the results from the studentized and BCa intervals tend to provide more accurate coverage rates.

Unless the standard deviation function for the studentized interval is input via the `sdfun` argument, the studentized interval can be quite computationally costly. This is because an inner bootstrap is needed to estimate the standard deviation of the statistic for each (outer) bootstrap replicate—and you may want to increase the default number of inner bootstrap replicates (see Note).

The efficiency of the BCa interval will depend on the sample size n and the computational complexity of the (jackknife) statistic estimate. Assuming that n is not too large and the jackknife statistic is not too difficult to compute, the BCa interval can be computed reasonably quickly—especially in comparison the studentized interval with an inner bootstrap.

Computational details of the various confidence intervals are described in Efron and Tibshirani (1994) and in Davison and Hinkley (1997). For a useful and concise discussion of the various intervals, see Carpenter and Bithell (2000).

Value

`t0`	Observed statistic, computed using `statistic(x,...)`
`se`	Bootstrap estimate of the standard error.
`bias`	Bootstrap estimate of the bias.
`cov`	Bootstrap estimate of the covariance (for multivariate statistics).
`normal`	Normal approximation confidence interval(s).
`basic`	Basic (reverse percentile) confidence interval(s).
`percent`	Percentile confidence interval(s).
`student`	Studentized (bootstrap-t) confidence interval(s).
`bca`	Bias-corrected and accelerated (BCa) confidence interval(s).
`z0`	Bias-correction factor(s). Only provided if `bca %in% method`.
`acc`	Acceleration factor(s). Only provided if `bca %in% method`.
`boot.dist`	Bootstrap distribution of statistic(s). Only provided if `boot.dist = TRUE`.

R	Number of bootstrap replicates (same as input).
level	Confidence level (same as input).
sdfun	Standard deviation function for statistic (same as input).
sdrep	Number of inner bootstrap replicates (same as input).
jackknife	Jackknife function (same as input).

Note

If `boot.dist = TRUE`, the output `boot.dist` will be a matrix of dimension `R` by `length(statistic(x,...))` if the statistic is multivariate. Otherwise the bootstrap distribution will be a vector of length `R`.

For the `"stud"` method, the default of `sdrep = 99` may produce a crude estimate of the standard deviation of the statistic(s). For more accurate estimates, the value of `sdrep` may need to be set substantially larger, e.g., `sdrep = 999`.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Carpenter, J., & Bithell, J. (2000). Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Statistics in Medicine, 19*(9), 1141-1164. doi: 10.1002/(SICI)1097-0258(20000515)19:9%3C1141::AID-SIM479%3E3.0.CO;2-F

Davison, A. C., & Hinkley, D. V. (1997). *Bootstrap Methods and Their Application*. Cambridge University Press. doi: 10.1017/CBO9780511802843

Efron, B., & Tibshirani, R. J. (1994). *An Introduction to the Boostrap*. Chapman & Hall/CRC. doi: 10.1201/9780429246593

Examples

```
######*******#######   UNIVARIATE DATA   ######*******#######

### Example 1: univariate statistic (median)

# generate 100 standard normal observations
set.seed(1)
n <- 100
x <- rnorm(n)

# nonparametric bootstrap
npbs <- np.boot(x = x, statistic = median)
npbs


### Example 2: multivariate statistic (quartiles)

# generate 100 standard normal observations
set.seed(1)
n <- 100
```

```
x <- rnorm(n)

# nonparametric bootstrap
npbs <- np.boot(x = x, statistic = quantile,
                probs = c(0.25, 0.5, 0.75))
npbs



######*******#######   MULTIVARIATE DATA   ######*******#######

### Example 1: univariate statistic (correlation)

# correlation matrix square root (with rho = 0.5)
rho <- 0.5
val <- c(sqrt(1 + rho), sqrt(1 - rho))
corsqrt <- matrix(c(val[1], -val[2], val), 2, 2) / sqrt(2)

# generate 100 bivariate observations (with rho = 0.5)
n <- 100
set.seed(1)
data <- cbind(rnorm(n), rnorm(n)) %*% corsqrt

# define statistic function
statfun <- function(x, data) cor(data[x,1], data[x,2])

# nonparametric bootstrap
npbs <- np.boot(x = 1:n, statistic = statfun, data = data)
npbs


### Example 2: multivariate statistic (variances and covariance)

# correlation matrix square root (with rho = 0.5)
rho <- 0.5
val <- c(sqrt(1 + rho), sqrt(1 - rho))
corsqrt <- matrix(c(val[1], -val[2], val), 2, 2) / sqrt(2)

# generate 100 bivariate observations (with rho = 0.5)
n <- 100
set.seed(1)
data <- cbind(rnorm(n), rnorm(n)) %*% corsqrt

# define statistic function
statfun <- function(x, data) {
  cmat <- cov(data[x,])
  ltri <- lower.tri(cmat, diag = TRUE)
  cvec <- cmat[ltri]
  names(cvec) <- c("var(x1)", "cov(x1,x2)", "var(x2)")
  cvec
}

# nonparametric bootstrap
```

```
npbs <- np.boot(x = 1:n, statistic = statfun, data = data)
npbs



## Not run:

#####***######   REGRESSION   #####***######

### Example 1: bootstrap cases

# generate 100 observations
n <- 100
set.seed(1)
x <- seq(0, 1, length.out = n)
y <- 1 + 2 * x + rnorm(n)
data <- data.frame(x = x, y = y)

# define statistic function
statfun <- function(x, data) {
  xmat <- cbind(1, data$x[x])
  xinv <- solve(crossprod(xmat)) %*% t(xmat)
  coef <- as.numeric(xinv %*% data$y[x])
  names(coef) <- c("(Intercept)", "x")
  coef
}

# nonparametric bootstrap
npbs <- np.boot(x = 1:n, statistic = statfun, data = data)
npbs


### Example 2: bootstrap residuals

# generate 100 observations
n <- 100
set.seed(1)
x <- seq(0, 1, length.out = n)
y <- 1 + 2 * x + rnorm(n)

# prepare data
xmat <- cbind(1, x)
xinv <- solve(crossprod(xmat)) %*% t(xmat)
fit <- xmat %*% xinv %*% y
data <- list(fit = fit, resid = y - fit, xinv = xinv, x = x)

# define statistic function
statfun <- function(x, data) {
  ynew <- data$fit + data$resid[x]
  coef <- as.numeric(data$xinv %*% ynew)
  names(coef) <- c("(Intercept)", "x")
  coef
}
```

```
# define jackknife function
jackfun <- function(x, data){
  ynew <- data$fit[x] + data$resid[x]
  xmat <- cbind(1, data$x[x])
  xinv <- solve(crossprod(xmat)) %*% t(xmat)
  coef <- as.numeric(xinv %*% ynew)
  names(coef) <- c("(Intercept)", "x")
  coef
}

# nonparametric bootstrap
npbs <- np.boot(x = 1:n, statistic = statfun, data = data,
                jackknife = jackfun)
npbs

## End(Not run)
```

`np.cor.test` *Nonparametric Tests of Correlation Coefficients*

Description

Denoting the Pearson product-moment correlation coefficient as

$$\rho = Cov(X, Y)/\sqrt{Var(X)Var(Y)}$$

this function implements permutation tests of $H_0 : \rho = \rho_0$ where ρ_0 is the user-specified null value. Can also implement tests of partial correlations, semi-partial (or part) correlations, and independence.

Usage

```
np.cor.test(x, y, z = NULL,
            alternative = c("two.sided", "less", "greater"),
            rho = 0, independent = FALSE, partial = TRUE,
            R = 9999, parallel = FALSE, cl = NULL,
            perm.dist = TRUE)
```

Arguments

`x`	X vector (n by 1).
`y`	Y vector (n by 1).
`z`	Optional Z matrix (n by q). If provided, the partial (or semi-partial if `partial = FALSE`) correlation is calculated between `x` and `y` controlling for `z`.
`alternative`	Alternative hypothesis. Must be either "two.sided" ($H_1 : \rho \neq \rho_0$), "less" ($H_1 : \rho < \rho_0$), or "greater" ($H_1 : \rho > \rho_0$).

`rho`	Null hypothesis value ρ_0. Defaults to zero.
`independent`	If `FALSE` (default), the null hypothesis is $H_0 : \rho = \rho_0$. Otherwise, the null hythpothesis is that X and Y are independent, i.e., $H_0 : F_{XY}(x, y) = F_X(x)F_Y(y)$.
`partial`	Only applicable if `z` is provided. If `TRUE` (default), the partial correlation between `x` and `y` controlling for `z` is tested. Otherwise the semi-partial correlation is tested. See Details.
`R`	Number of resamples for the permutation test (positive integer).
`parallel`	Logical indicating if the `parallel` package should be used for parallel computing (of the permutation distribution). Defaults to `FALSE`, which implements sequential computing.
`cl`	Cluster for parallel computing, which is used when `parallel = TRUE`. Note that if `parallel = TRUE` and `cl = NULL`, then the cluster is defined as `makeCluster(detectCores())`.
`perm.dist`	Logical indicating if the permutation distribution should be returned.

Details

Default use of this function tests the Pearson correlation between X and Y using the studentized test statistic proposed by DiCiccio and Romano (2017). If `independent = TRUE`, the classic (unstudentized) test statistic is used to test the null hypothesis of independence.

If Z is provided, the partial or semi-partial correlation between X and Y controlling for Z is tested. For the semi-partial correlation, the effect of Z is partialled out of X.

Value

`statistic`	Test statistic value.
`p.value`	p-value for testing $H_0 : \rho = \rho_0$ or $H_0 : F_{XY}(x, y) = F_X(x)F_Y(y)$.
`perm.dist`	Permutation distribution of `statistic`.
`alternative`	Alternative hypothesis.
`null.value`	Null hypothesis value for ρ.
`independent`	Independence test?
`R`	Number of resamples.
`exact`	Exact permutation test? See Note.
`estimate`	Sample estimate of correlation coefficient ρ.

Note

The permutation test will be exact when the requested number of resamples `R` is greater than `factorial(n)` minus one. In this case, the permutation distribution `perm.dist` contains all `factorial(n)` possible values of the test statistic.

If `z = NULL`, the result will be the same as using `np.reg.test` with `method = "perm"`.

If `z` is supplied and `partial = TRUE`, the result will be the same as using `np.reg.test` with `method = "KC"` and `homosced = FALSE`.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

DiCiccio, C. J., & Romano, J. P. (2017). Robust permutation tests for correlation and regression coefficients. Journal of the American Statistical Association, 112(519), 1211-1220. doi: 10.1080/01621459.2016.1202117

Helwig, N. E. (2019). Statistical nonparametric mapping: Multivariate permutation tests for location, correlation, and regression problems in neuroimaging. WIREs Computational Statistics, 11(2), e1457. doi: 10.1002/wics.1457

Pitman, E. J. G. (1937b). Significance tests which may be applied to samples from any populations. ii. the correlation coefficient test. Supplement to the Journal of the Royal Statistical Society, 4(2), 225-232. doi: 10.2307/2983647

See Also

`plot.np.cor.test` S3 plotting method for visualizing the results

Examples

```
# generate data
rho <- 0.5
val <- c(sqrt(1 + rho), sqrt(1 - rho))
corsqrt <- matrix(c(val[1], -val[2], val), 2, 2) / sqrt(2)
set.seed(1)
n <- 10
z <- cbind(rnorm(n), rnorm(n)) %*% corsqrt
x <- z[,1]
y <- z[,2]

# test H0: rho = 0
set.seed(0)
np.cor.test(x, y)

# test H0: X and Y are independent
set.seed(0)
np.cor.test(x, y, independent = TRUE)
```

`np.loc.test` *Nonparametric Tests of Location Parameters*

Description

Performs one and two sample nonparametric (randomization) tests of location parameters, i.e., means and medians. Implements univariate and multivariate tests using eight different test statistics: Student's one-sample t-test, Johnson's modified t-test, Wilcoxon's Signed Rank test, Fisher's Sign test, Student's two-sample t-test, Welch's t-test, Wilcoxon's Rank Sum test (i.e., Mann-Whitney's U test), and a studentized Wilcoxon test for unequal variances.

Usage

```
np.loc.test(x, y = NULL,
            alternative = c("two.sided", "less", "greater"),
            mu = 0, paired = FALSE, var.equal = FALSE,
            median.test = FALSE, symmetric = TRUE,
            R = 9999, parallel = FALSE, cl = NULL,
            perm.dist = TRUE)
```

Arguments

`x`	Numeric vector (or matrix) of data values.
`y`	Optional numeric vector (or matrix) of data values.
`alternative`	Alternative hypothesis. Must be either "two.sided" ($H_1 : \mu \neq \mu_0$), "less" ($H_1 : \mu < \mu_0$), or "greater" ($H_1 : \mu > \mu_0$).
`mu`	Null hypothesis value μ_0. Defaults to zero.
`paired`	Logical indicating whether you want a paired location test.
`var.equal`	Logical indicating whether to treat the two variances as being equal.
`median.test`	Logical indicating whether the location test is for the median. Default is `FALSE`, i.e., μ is the mean.
`symmetric`	Logical indicating if the distribution of `x` should be assumed to be symmetric around μ. Only used for one (or paired) sample tests.
`R`	Number of resamples for the permutation test (positive integer).
`parallel`	Logical indicating if the `parallel` package should be used for parallel computing (of the permutation distribution). Defaults to `FALSE`, which implements sequential computing.
`cl`	Cluster for parallel computing, which is used when `parallel = TRUE`. Note that if `parallel = TRUE` and `cl = NULL`, then the cluster is defined as `makeCluster(detectCores())`.
`perm.dist`	Logical indicating if the permutation distribution should be returned.

Details

One sample	μ is the mean (or median) of X
Paired	μ is the mean (or median) of $X - Y$
Two sample	μ is the mean difference $E(X) - E(Y)$
	or the median of the differences $X - Y$

For one (or paired) sample tests, the different test statistics can be obtained using

	`median.test = F`	`median.test = T`
`symmetric = F`	Johnson t test	Fisher sign test
`symmetric = T`	Student t test	Wilcoxon signed rank test

For two sample tests, the different test statistics can be obtained using

	`median.test = F`	`median.test = T`
`var.equal = F`	Welch t test	Studentized Wilcoxon test
`var.equal = T`	Student t test	Wilcoxon rank sum test

Value

- `statistic` Test statistic value.
- `p.value` p-value for testing $H_0 : \mu = \mu_0$.
- `perm.dist` Permutation distribution of `statistic`.
- `alternative` Alternative hypothesis.
- `null.value` Null hypothesis value for μ.
- `var.equal` Assuming equal variances? Only for two sample tests.
- `median.test` Testing the median?
- `symmetric` Assuming symmetry? Only for one sample and paired tests.
- `R` Number of resamples.
- `exact` Exact permutation test? See Note.
- `estimate` Estimate of parameter μ.
- `univariate` Univariate test statistic value for j-th variable (for multivariate input).
- `adj.p.value` Adjusted p-value for testing significance of j-th variable (for multivariate input).
- `method` Method used for permutation test. See Details.

Multivariate Tests

If the input `x` (and possibly `y`) is a matrix with $m > 1$ columns, the multivariate test statistic is defined as

`alternative`	`statistic`
`two.sided`	`max(abs(univariate))`
`less`	`min(univariate)`
`greater`	`max(univariate)`

The global null hypothesis (across all m variables) is tested by comparing the observed `statistic` to the permutation distribution `perm.dist`. This produces the `p.value` for testing the global null hypothesis.

The local null hypothesis (separately for each variable) is tested by comparing the `univariate` test statistic to `perm.dist`. This produces the adjusted p-values (`adj.p.values`), which control the familywise Type I error rate across the m tests.

Note

For one sample (or paired) tests, the permutation test will be exact when the requested number of resamples `R` is greater than `2^n` minus one. In this case, the permutation distribution `perm.dist` contains all `2^n` possible values of the test statistic.

For two sample tests, the permutation test will be exact when the requested number of resamples `R` is greater than `choose(N,n)` minus one, where `m = length(x)`, `n = length(y)`, and `N = m + n`. In this case, the permutation distribution `perm.dist` contains all `choose(N,n)` possible values of the test statistic.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Blair, R. C., Higgins, J. J., Karniski, W., & Kromrey, J. D. (1994). A study of multivariate permutation tests which may replace Hotelling's T2 test in prescribed circumstances. Multivariate Behavioral Research, 29(2), 141-163. doi: 10.1207/s15327906mbr2902_2

Chung, E., & Romano, J. P. (2016). Asymptotically valid and exact permutation tests based on two-sample U-statistics. Journal of Statistical Planning and Inference, 168, 97-105. doi: 10.1016/j.jspi.2015.07.004

Fisher, R. A. (1925). Statistical methods for research workers. Edinburgh: Oliver and Boyd.

Helwig, N. E. (2019). Statistical nonparametric mapping: Multivariate permutation tests for location, correlation, and regression problems in neuroimaging. WIREs Computational Statistics, 11(2), e1457. doi: 10.1002/wics.1457

Janssen, A. (1997). Studentized permutation tests for non-i.i.d. hypotheses and the generalized Behrens-Fisher problem. Statistics & Probability Letters , 36 (1), 9-21. doi: 10.1016/S0167-7152(97)00043-6

Johnson, N. J. (1978). Modified t tests and confidence intervals for asymmetrical populations. Journal of the American Statistical Association, 73 (363), 536-544. doi: 10.2307/2286597

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. Annals Of Mathematical Statistics, 18(1), 50-60. doi: 10.1214/aoms/1177730491

Pitman, E. J. G. (1937a). Significance tests which may be applied to samples from any populations. Supplement to the Journal of the Royal Statistical Society, 4(1), 119-130. doi: 10.2307/2984124

Romano, J. P. (1990). On the behavior of randomization tests without a group invariance assumption. Journal of the American Statistical Association, 85(411), 686-692. doi: 10.1080/01621459.1990.10474928

Student. (1908). The probable error of a mean. Biometrika, 6(1), 1-25. doi: 10.2307/2331554

Welch, B. L. (1938). The significance of the difference between two means when the population variances are unequal. Biometrika, 39(3/4), 350-362. doi: 10.2307/2332010

Wilcoxon, F. (1945). Individual comparisons by ranking methods. Biometrics Bulletin, 1(6), 80-83. doi: 10.2307/3001968

See Also

plot.np.loc.test S3 plotting method for visualizing the results

Examples

```
#####*********#####   UNIVARIATE   #####*********#####

###***###   ONE SAMPLE   ###***###

# generate data
set.seed(1)
n <- 10
x <- rnorm(n, mean = 0.5)

# one sample t-test
set.seed(0)
np.loc.test(x)

# Johnson t-test
set.seed(0)
np.loc.test(x, symmetric = FALSE)

# Wilcoxon signed rank test
set.seed(0)
np.loc.test(x, median.test = TRUE)

# Fisher sign test
set.seed(0)
np.loc.test(x, median.test = TRUE, symmetric = FALSE)


###***###   PAIRED SAMPLE   ###***###

# generate data
set.seed(1)
n <- 10
x <- rnorm(n, mean = 0.5)
y <- rnorm(n)

# paired t-test
set.seed(0)
np.loc.test(x, y, paired = TRUE)

# paired Johnson t-test
set.seed(0)
np.loc.test(x, y, paired = TRUE, symmetric = FALSE)

# paired Wilcoxon signed rank test
set.seed(0)
np.loc.test(x, y, paired = TRUE, median.test = TRUE)
```

```
# paired Fisher sign test
set.seed(0)
np.loc.test(x, y, paired = TRUE, median.test = TRUE, symmetric = FALSE)


######***###   TWO SAMPLE   ######***###

# generate data
set.seed(1)
m <- 7
n <- 8
x <- rnorm(m, mean = 0.5)
y <- rnorm(n)

# Welch t-test
set.seed(0)
np.loc.test(x, y)

# Student t-test
set.seed(0)
np.loc.test(x, y, var.equal = TRUE)

# Studentized Wilcoxon test
set.seed(0)
np.loc.test(x, y, median.test = TRUE)

# Wilcoxon rank sum test
set.seed(0)
np.loc.test(x, y, var.equal = TRUE, median.test = TRUE)



## Not run:

#####***********######   MULTIVARIATE   #####*********######

######***###   ONE SAMPLE   ######***###

# generate data
set.seed(1)
n <- 10
x <- cbind(rnorm(n, mean = 0.5),
           rnorm(n, mean = 1),
           rnorm(n, mean = 1.5))

# multivariate one sample t-test
set.seed(0)
ptest <- np.loc.test(x)
ptest
ptest$univariate
ptest$adj.p.values
```

```
#########   PAIRED SAMPLE   #########

# generate data
set.seed(1)
n <- 10
x <- cbind(rnorm(n, mean = 0.5),
           rnorm(n, mean = 1),
           rnorm(n, mean = 1.5))
y <- matrix(rnorm(n * 3), nrow = n, ncol = 3)

# multivariate paired t-test
set.seed(0)
ptest <- np.loc.test(x, y, paired = TRUE)
ptest
ptest$univariate
ptest$adj.p.values


#########   TWO SAMPLE   #########

# generate data
set.seed(1)
m <- 7
n <- 8
x <- cbind(rnorm(m, mean = 0.5),
           rnorm(m, mean = 1),
           rnorm(m, mean = 1.5))
y <- matrix(rnorm(n * 3), nrow = n, ncol = 3)

# multivariate Welch t-test
set.seed(0)
ptest <- np.loc.test(x, y)
ptest
ptest$univariate
ptest$adj.p.values


## End(Not run)
```

`np.reg.test` *Nonparametric Tests of Regression Coefficients*

Description

Assuming a linear model of the form

$$Y = \alpha + X\beta + \epsilon$$

or

$$Y = \alpha + X\beta + Z\gamma + \epsilon$$

this function implements permutation tests of $H_0 : \beta = \beta_0$ where β_0 is the user-specified null vector.

Usage

```
np.reg.test(x, y, z = NULL, method = NULL,
            beta = NULL, homosced = FALSE, lambda = 0,
            R = 9999, parallel = FALSE, cl = NULL,
            perm.dist = TRUE)
```

Arguments

`x`	Matrix of predictor variables (`n` by `p`).
`y`	Response vector or matrix (`n` by `m`).
`z`	Optional matrix of nuisance variables (`n` by `q`).
`method`	Permutation method. See Details.
`beta`	Null hypothesis value for β (`p` by `m`). Defaults to matrix of zeros.
`homosced`	Are the ϵ terms homoscedastic? If `FALSE` (default), a robust Wald test statistic is used. Otherwise the classic F test statistic is used.
`lambda`	Scalar or vector of ridge parameter(s). Defaults to vector of zeros.
`R`	Number of resamples for the permutation test (positive integer).
`parallel`	Logical indicating if the `parallel` package should be used for parallel computing (of the permutation distribution). Defaults to `FALSE`, which implements sequential computing.
`cl`	Cluster for parallel computing, which is used when `parallel = TRUE`. Note that if `parallel = TRUE` and `cl = NULL`, then the cluster is defined as `makeCluster(detectCores())`.
`perm.dist`	Logical indicating if the permutation distribution should be returned.

Details

With no nuisance variables in the model (i.e., `z = NULL`), there are three possible options for the `method` argument:

Method	Model
perm	$PY = \alpha + X\beta + \epsilon$
flip	$SY = \alpha + X\beta + \epsilon$
both	$PSY = \alpha + X\beta + \epsilon$

where P is a permutation matrix and S is a sign-flipping matrix.

With nuisance variables in the model, there are eight possible options for the `method` argument:

Method	Name	Model
HJ	Huh-Jhun	$PQ'R_zY = \alpha + Q'R_zX\beta + \epsilon$
KC	Kennedy-Cade	$PR_zY = \alpha + R_zX\beta + \epsilon$
SW	Still-White	$PR_zY = \alpha + X\beta + \epsilon$
TB	ter Braak	$(PR_m + H_m)Y = \alpha + X\beta + Z\gamma + \epsilon$

FL	Freedman-Lane	$(PR_z + H_z)Y = \alpha + X\beta + Z\gamma + \epsilon$
MA	Manly	$PY = \alpha + X\beta + Z\gamma + \epsilon$
OS	O'Gorman-Smith	$Y = \alpha + PR_zX\beta + Z\gamma + \epsilon$
DS	Draper-Stoneman	$Y = \alpha + PX\beta + Z\gamma + \epsilon$

where P is permutation matrix and Q is defined as $R_z = QQ'$ with $Q'Q = I$.

Note that H_z is the hat matrix for the nuisance variable design matrix, and $R_z = I - H_z$ is the corresponding residual forming matrix. Similarly, H_m and R_m are the hat and residual forming matrices for the full model including the predictor and nuisance variables.

Value

`statistic`	Test statistic value.
`p.value`	p-value for testing $H_0 : \beta = \beta_0$.
`perm.dist`	Permutation distribution of `statistic`.
`method`	Permutation method.
`null.value`	Null hypothesis value for β.
`homosced`	Homoscedastic errors?
`R`	Number of resamples.
`exact`	Exact permutation test? See Note.
`coefficients`	Least squares estimates of α, β, and γ (if applicable).
`univariate`	Univariate test statistic value for j-th variable (for multivariate inputs).
`adj.p.value`	Adjusted p-value for testing significance of j-th variable (for multivariate inputs).

Multivariate Tests

If the input `y` is a matrix with $m > 1$ columns, the multivariate test statistic is defined as `statistic = max(univariate)` given that the `univariate` test statistics are non-negative.

The global null hypothesis (across all m variables) is tested by comparing the observed `statistic` to the permutation distribution `perm.dist`. This produces the `p.value` for testing the global null hypothesis.

The local null hypothesis (separately for each variable) is tested by comparing the `univariate` test statistic to `perm.dist`. This produces the adjusted p-values (`adj.p.values`), which control the familywise Type I error rate across the m tests.

Note

If `method = "flip"`, the permutation test will be exact when the requested number of resamples `R` is greater than `2^n` minus one. In this case, the permutation distribution `perm.dist` contains all `2^n` possible values of the test statistic.

If `method = "both"`, the permutation test will be exact when the requested number of resamples `R` is greater than `factorial(n) * (2^n)` minus one. In this case, the permutation distribution `perm.dist` contains all `factorial(n) * (2^n)` possible values of the test statistic.

If `method = "HJ"`, the permutation test will be exact when the requested number of resamples `R` is greater than `factorial(n-q-1)` minus one. In this case, the permutation distribution `perm.dist` contains all `factorial(n-q-1)` possible values of the test statistic.

Otherwise the permutation test will be exact when the requested number of resamples `R` is greater than `factorial(n)` minus one. In this case, the permutation distribution `perm.dist` contains all `factorial(n)` possible values of the test statistic.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

DiCiccio, C. J., & Romano, J. P. (2017). Robust permutation tests for correlation and regression coefficients. Journal of the American Statistical Association, 112(519), 1211-1220. doi: 10.1080/01621459.2016.1202117

Draper, N. R., & Stoneman, D. M. (1966). Testing for the inclusion of variables in linear regression by a randomisation technique. Technometrics, 8(4), 695-699. doi: 10.2307/1266641

Freedman, D., & Lane, D. (1983). A nonstochastic interpretation of reported significance levels. Journal of Business and Economic Statistics, 1(4), 292-298. doi: 10.2307/1391660

Helwig, N. E. (2019a). Statistical nonparametric mapping: Multivariate permutation tests for location, correlation, and regression problems in neuroimaging. WIREs Computational Statistics, 11(2), e1457. doi: 10.1002/wics.1457

Helwig, N. E. (2019b). Robust nonparametric tests of general linear model coefficients: A comparison of permutation methods and test statistics. NeuroImage, 201, 116030. doi: 10.1016/j.neuroimage.2019.116030

Huh, M.-H., & Jhun, M. (2001). Random permutation testing in multiple linear regression. Communications in Statistics - Theory and Methods, 30(10), 2023-2032. doi: 10.1081/STA-100106060

Kennedy, P. E., & Cade, B. S. (1996). Randomization tests for multiple regression. Communications in Statistics - Simulation and Computation, 25(4), 923-936. doi: 10.1080/03610919608813350

Manly, B. (1986). Randomization and regression methods for testing for associations with geographical, environmental and biological distances between populations. Researches on Population Ecology, 28(2), 201-218. doi: 10.1007/BF02515450

Nichols, T. E., Ridgway, G. R., Webster, M. G., & Smith, S. M. (2008). GLM permutation: nonparametric inference for arbitrary general linear models. NeuroImage, 41(S1), S72.

O'Gorman, T. W. (2005). The performance of randomization tests that use permutations of independent variables. Communications in Statistics - Simulation and Computation, 34(4), 895-908. doi: 10.1080/03610910500308230

Still, A. W., & White, A. P. (1981). The approximate randomization test as an alternative to the F test in analysis of variance. British Journal of Mathematical and Statistical Psychology, 34(2), 243-252. doi: 10.1111/j.2044-8317.1981.tb00634.x

ter Braak, C. J. F. (1992). Permutation versus bootstrap significance tests in multiple regression and ANOVA. In K. H. J\"ockel, G. Rothe, & W. Sendler (Eds.), Bootstrapping and related techniques. lecture notes in economics and mathematical systems, vol 376 (pp. 79-86). Springer.

White, H. (1980). A heteroscedasticity-consistent covariance matrix and a direct test for heteroscedasticity. Econometrica, 48(4), 817-838. doi: 10.2307/1912934

Winkler, A. M., Ridgway, G. R., Webster, M. A., Smith, S. M., & Nichols, T. E. (2014). Permutation inference for the general linear model. NeuroImage, 92, 381-397. doi: 10.1016/j.neuroimage.2014.01.060

See Also

`plot.np.reg.test` S3 plotting method for visualizing the results

Examples

```
######*********######   UNIVARIATE   ######*********######

###***###   TEST ALL COEFFICIENTS   ###***###

# generate data
set.seed(1)
n <- 10
x <- cbind(rnorm(n), rnorm(n))
y <- rnorm(n)

# Wald test (method = "perm")
set.seed(0)
np.reg.test(x, y)

# F test (method = "perm")
set.seed(0)
np.reg.test(x, y, homosced = TRUE)


###***###   TEST SUBSET OF COEFFICIENTS   ###***###

# generate data
set.seed(1)
n <- 10
x <- rnorm(n)
z <- rnorm(n)
y <- 3 + 2 * z + rnorm(n)

# Wald test (method = "HJ")
set.seed(0)
np.reg.test(x, y, z)

# F test (method = "HJ")
set.seed(0)
np.reg.test(x, y, z, homosced = TRUE)


## Not run:

######*********######   MULTIVARIATE   ######*********######

###***###   TEST ALL COEFFICIENTS   ###***###
```

```
# generate data
set.seed(1)
n <- 10
x <- cbind(rnorm(n), rnorm(n))
y <- matrix(rnorm(n * 3), nrow = n, ncol = 3)

# multivariate Wald test (method = "perm")
set.seed(0)
np.reg.test(x, y)

# multivariate F test (method = "perm")
set.seed(0)
np.reg.test(x, y, homosced = TRUE)


######***###   TEST SUBSET OF COEFFICIENTS   ###***###

# generate data
set.seed(1)
n <- 10
x <- rnorm(n)
z <- rnorm(n)
y <- cbind(1 + 3 * z + rnorm(n),
           2 + 2 * z + rnorm(n),
           3 + 1 * z + rnorm(n))

# multivariate Wald test (method = "HJ")
set.seed(0)
np.reg.test(x, y, z)

# multivariate F test (method = "HJ")
set.seed(0)
np.reg.test(x, y, z, homosced = TRUE)


## End(Not run)
```

`permn` *Generate All Permutations of n Elements*

Description

Generates all $n!$ vectors of length n consisting of permutations of the integers 1 to n.

Usage

```
permn(n)
```

Arguments

`n`	Number of elements.

Details

Adapted from the "permutations" function in the e1071 R package.

Value

Matrix of dimension n by $n!$ where each column contains a unique permutation vector.

Warning

For large `n` this function will consume a lot of memory and may even crash R.

Note

Used for exact tests in `np.cor.test` and `np.reg.test`.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2018). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-0. https://CRAN.R-project.org/package=e1071

Examples

```
permn(2)
permn(3)
```

`plot` *Plots Permutation Distribution for Nonparametric Tests*

Description

`plot` methods for object classes "np.cor.test", "np.loc.test", and "np.reg.test"

Usage

```
## S3 method for class 'np.cor.test'
plot(x, alpha = 0.05, col = "grey", col.rr = "red",
     col.stat = "black", lty.stat = 2, lwd.stat = 2,
     xlab = "Test Statistic", main = "Permutation Distribution",
     breaks = "scott", border = NA, box = TRUE, ...)

## S3 method for class 'np.loc.test'
plot(x, alpha = 0.05, col = "grey", col.rr = "red",
     col.stat = "black", lty.stat = 2, lwd.stat = 2,
     xlab = "Test Statistic", main = "Permutation Distribution",
     breaks = "scott", border = NA, box = TRUE, ...)

## S3 method for class 'np.reg.test'
plot(x, alpha = 0.05, col = "grey", col.rr = "red",
     col.stat = "black", lty.stat = 2, lwd.stat = 2,
     xlab = "Test Statistic", main = "Permutation Distribution",
     breaks = "scott", border = NA, box = TRUE, SQRT = TRUE, ...)
```

Arguments

`x`	an object of class "np.cor.test" output by the `np.cor.test` function, "np.loc.test" output by the `np.loc.test` function, or "np.reg.test" output by the `np.reg.test` function
`alpha`	significance level of the nonparametric test
`col`	color for plotting the non-rejection region
`col.rr`	color for plotting the rejection region
`col.stat`	color for plotting the observed test statistic
`lty.stat`	line type for plotting the observed test statistic
`lwd.stat`	line width for plotting the observed test statistic
`xlab`	x-axis label for the plot
`main`	title for the plot
`breaks`	defines the breaks of the histogram (see `hist`)
`border`	color of the border around the bars
`box`	should a box be drawn around the plot?
`SQRT`	for regression tests, should the permutation distribution (and test statistic) be plotted on the square-root scale?
`...`	additional arguments to be passed to `hist`

Details

Plots a histogram of the permutation distribution and the observed test statistic. The argument 'alpha' controls the rejection region of the nonparametric test, which is plotted using a separate color (default is red).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E. (2019). Statistical nonparametric mapping: Multivariate permutation tests for location, correlation, and regression problems in neuroimaging. WIREs Computational Statistics, 11(2), e1457. doi: 10.1002/wics.1457

See Also

np.cor.test for information on nonparametric correlation tests

np.loc.test for information on nonparametric location tests

np.reg.test for information on nonparametric regression tests

Examples

```
######***********######   np.cor.test   ######***********######

# generate data
rho <- 0.5
val <- c(sqrt(1 + rho), sqrt(1 - rho))
corsqrt <- matrix(c(val[1], -val[2], val), 2, 2) / sqrt(2)
set.seed(1)
n <- 50
z <- cbind(rnorm(n), rnorm(n)) %*% corsqrt
x <- z[,1]
y <- z[,2]

# test H0: rho = 0
set.seed(0)
test <- np.cor.test(x, y)

# plot results
plot(test)


######***********######   np.loc.test   ######***********######

# generate data
set.seed(1)
n <- 50
x <- rnorm(n, mean = 0.5)

# one sample t-test
set.seed(0)
test <- np.loc.test(x)

# plot results
plot(test)
```

```
#####**********######   np.reg.test   #####**********######

# generate data
set.seed(1)
n <- 50
x <- cbind(rnorm(n), rnorm(n))
beta <- c(0.25, 0.5)
y <- x %*% beta + rnorm(n)

# Wald test (method = "perm")
set.seed(0)
test <- np.reg.test(x, y)

# plot results
plot(test)
```

Index