Package 'robusTest'

June 18, 2024

Title Calibrated Correlation and Two-Sample Tests

Version 1.1.0

Description Implementation of corrected two-
sample tests. A corrected version of the Pearson and Kendall correlation tests,
the Mann-Whitney (Wilcoxon) rank sum test, the Wilcoxon signed rank test and a variance test are implemented.
The package also proposes a test for the median and an independence test between two continuous variables of Kolmogorov-Smirnov's type.
All these corrected tests are asymptotically calibrated in the sense that the probability of rejection under the null hypothesis is
asymptotically equal to the level of the test. See <doi:10.48550/arXiv.2211.08784> for more details on the statistical tests.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 2.10)

LinkingTo Rcpp

Imports Rcpp, stats

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Olivier Bouaziz [aut, cre],
Jérôme Dedecker [aut],
Anatole Dedecker [aut]

Maintainer Olivier Bouaziz <olivier.bouaziz@parisdescartes.fr>

Repository CRAN

Date/Publication 2024-06-18 11:00:02 UTC

Contents

cortest . . . 2
ecdf2D . . . 5
Evans . . . 6
indeptest . . . 7
mediantest . . . 9
simulecdf . . . 11
stat_indeptest . . . 12
tiebreak . . . 13
vartest . . . 14
wilcoxtest . . . 15

Index **19**

`cortest` *Calibrated tests for correlation between paired samples*

Description

Tests the association/correlation for continuous paired samples using corrected versions of Pearson's, Kendall's and Spearman's correlation tests. These three tests are asymptotically well calibrated.

Usage

```
cortest(
  x,
  y,
  alternative = "two.sided",
  method = "pearson",
  ties.break = "none",
  conf.level = 0.95
)
```

Arguments

`x, y`	the two continuous variables. Must be of same length.
`alternative`	indicates the alternative hypothesis and must be one of "two.sided", "greater" or "less".
`method`	a character string indicating which test to implement. Can be `"pearson"`, `"kendall"` or `"spearman"`.
`ties.break`	the method used to break ties in case there are ties in the x or y vectors. Can be `"none"` or `"random"`.
`conf.level`	confidence level for the confidence interval of the correlation coefficient. It is used only for the Pearson's correlation test.

Details

Three tests are implemented. The null hypothesis for the corrected Pearson test is: H0 Cor(X,Y)=0 where Cor represents Pearson's correlation coefficient. The alternative is specified by the `alternative` argument. The null hypothesis for the corrected Kendall test is: H0 tau=0 where tau represents Kendall's tau coefficient. The null hypothesis for the corrected Spearman test is: H0 rho=0 where rho represents Spearman's rho coefficient. All tests are asymptotically well calibrated in the sense that the rejection probability under the null hypothesis is asymptotically equal to the level of the test. For the Pearson test, the exact distribution of the test statistic under the Gaussian case has been tabulated for n<130. For n>=130, the Student distribution with n-2 degrees of freedom is used.

When Pearson's correlation test is used, a confidence interval for Pearson's correlation coefficient is also returned. This confidence interval has been implemented from the delta-method. It should be noted that this method is asymptotic and can display very narrow intervals for small sample sizes and thus can suffer from low coverage probabilities. We therefore recommend to use confidence intervals for Pearson's correlation coefficient only when n is at least larger than 100.

The Kendall and Spearman correlation tests are not valid in the presence of ties in the `x` or `y` vector. If ties occur, they should be broken using the `ties.break` option. Note that they can also be broken outside `cortest` using the function `tiebreak`.

Value

Returns the result of the test with its corresponding p-value, the value of the test statistic and the estimated value of Pearson's correlation coefficient, Kendall's tau or Spearman's rho. For Pearson's correlation test an asymptotic confidence interval for the correlation coefficient is also returned.

Note

The option `ties.break` handles ties for both Kendall's and Spearman's test. If `ties.break="none"` the ties are ignored, if `ties.break="random"` they are randomly broken. Note that only ties inside each vector are broken (but not ties between the two vectors).

See Also

`vartest`, `indeptest`, `mediantest`, `wilcoxtest`, `tiebreak`.

Examples

```
#Application on the Evans dataset
#Description of this dataset is available in the lbreg package
data(Evans)
with(Evans,cor.test(CHL[CDH==1],DBP[CDH==1]))
with(Evans,cortest(CHL[CDH==1],DBP[CDH==1]))
#The pvalues are very different!

with(Evans,cortest(CHL[CDH==1],DBP[CDH==1],method="kendall",ties.break="random"))
with(Evans,cortest(CHL[CDH==1],DBP[CDH==1],method="spearman",ties.break="random"))

#We use the function tiebreak to remove ties and compare the results from cor.test with cortest
X=tiebreak(Evans$CHL[Evans$CDH==1])
Y=tiebreak(Evans$DBP[Evans$CDH==1])
```

```
cor.test(X,Y,method="kendall")
cortest(X,Y,method="kendall")
cor.test(X,Y,method="spearman")
cortest(X,Y,method="spearman")


#Simulated data
n=100 #sample size
M=10000 #number of replications
testone=function(n){
X=rnorm(n,0,1)
epsi=rnorm(n,0,1)
Y=X^2+0.3*epsi
list(test1=cortest(X,Y)$p.value,test2=cor.test(X,Y)$p.value) #cor.test is the standard Pearson test
}
res1=res2=rep(NA,M)
# Replications in order to check if the the corrected Pearson test and
# the standard test are well calibrated
for (i in 1:M)
{
result=testone(n)
res1[i]=result$test1
res2[i]=result$test2
}
mean(res1<0.05)
mean(res2<0.05)


#Replications with Kendall's test (may take a few minutes to run)
M=500
testone=function(n){
X=rnorm(n,0,1)
epsi=rnorm(n,0,1)
Y=X^2+0.3*epsi
list(test1=cortest(X,Y)$p.value,test2=cor.test(X,Y)$p.value,
test3=cortest(X,Y,method="kendall")$p.value,
test4=cor.test(X,Y,method="kendall")$p.value,
test5=cortest(X,Y,method="spearman")$p.value,
test6=cor.test(X,Y,method="spearman")$p.value)
#cor.test is the standard Pearson, Kendall or Spearman correlation test
}
res1=res2=res3=res4=res5=res6=rep(NA,M)
# Replications to check if the tests are well calibrated
for (i in 1:M)
{
result=testone(n)
res1[i]=result$test1
res2[i]=result$test2
res3[i]=result$test3
res4[i]=result$test4
res5[i]=result$test5
res6[i]=result$test6
}
```

```
mean(res1<0.05)
mean(res2<0.05)
mean(res3<0.05)
mean(res4<0.05)
mean(res5<0.05)
mean(res6<0.05)
```

---

ecdf2D *Bidimensional Empirical Cumulative Distribution Function*

Description

Compute the empirical cumulative distribution function for a bivariate continuous distribution.

Usage

```
ecdf2D(x, y)
```

Arguments

x, y	the two continuous variables. Must be of same length.

Details

The bidimensional e.c.d.f. (empirical cumulative distribution function) Fn is a step function with jumps i/n at observation values, where i is the number of tied observations at that value.

For observations (x1,y1), ..., (x_n,y_n), Fn is defined as

$$Fn(t1, t2) = 1/nsum_{i=1}^{n} Indicator(xi <= t1, yi <= t2)$$

Value

The result is returned as a matrix of dimension (n*n) where the entry (i,j) corresponds to Fn(xi,yj), i=1, ...,n, j=1, ...,n.

Note

Missing values are removed such that if a value of x (resp. y) is missing then the corresponding values of both x and y are removed. The bidimensional e.c.d.f. is then computed on the remaining elements.

See Also

indeptest; the bivariate package also provides plots for the bidimensional e.c.d.f.

Examples

```
#Simulated data #1
x<-c(0.2, 0.3, 0.1, 0.4)
y<-c(0.5, 0.4, 0.05, 0.2)
ecdf2D(x,y)

#Simulated data #2
n<-40
x<-rnorm(n)
y<-x^2+0.3*rnorm(n)
plot(x,y)
ecdf2D(x,y)
```

`Evans` *Evans County dataset*

Description

Data from cohort study in which white males in Evans County were followed for seven years, with coronary heart disease as the outcome of interest. This dataset comes from the `lbref` package.

Usage

```
Evans
```

Format

A data frame with 609 rows and 9 variables:

CDH outcome variable; 1 = coronary disease

CAT 1 = high, 0 = normal catecholamine level

AGE age (in years)

CHL cholesterol, mg/dl

SMK 1 = subject has ever smoked

ECG 1 = presence of electrocardiogram abnormality

DBP diastolic blood pressure, mmHg

SBP systolic blood pressure, mmHg

HPT 1 = SBP greater than or equal to 160 or DBP greater than or equal to 95

`indeptest` *Robust independence test for two continuous variables of Kolmogorov-Smirnov's type*

Description

Test the independence between two continuous variables based on the maximum distance between the joint empirical cumulative distribution function and the product of the marginal empirical cumulative distribution functions.

Usage

```
indeptest(
  x,
  y,
  N = 50000,
  simu = FALSE,
  ties.break = "none",
  nb_tiebreak = 100
)
```

Arguments

`x, y`	the two continuous variables. Must be of same length.
`N`	the number of Monte-Carlo replications if simu=TRUE.
`simu`	if TRUE a Monte-Carlo simulation with `N` replications is used to determine the distribution of the test statistic under the null hypothesis. If FALSE, pre computed tables are used (see Details for more information).
`ties.break`	the method used to break ties in case there are ties in the x or y vectors. Can be `"none"`, `"random"` or `"rep_random"`.
`nb_tiebreak`	the number for repetition for breaking the ties when `ties.break="rep_random"`.

Details

For two continuous variables, `indeptest` tests H0 X and Y are independent against H1 X and Y are not independent.

For observations (x1,y1), ..., (x_n,y_n), the bivariate e.c.d.f. (empirical cumulative distribution function) Fn is defined as:

$$Fn(t1, t2) = sum_{i=1}^{n} Indicator(xi <= t1, yi <= t2)/n.$$

Let Fn(t1) and Fn(t2) be the marginals e.c.d.f. The test statistic is defined as:

$$n^(1/2) sup_{t1,t2} |Fn(t1, t2) - Fn(t1) * Fn(t2)|.$$

Under H0 the test statistic is distribution free and is equivalent to the same test statistic computed for two independent continuous uniform variables in $[0, 1]$, where the supremum is taken for t1,t2

in $[0, 1]$. Using this result, the distribution of the test statistic is obtained using Monte-Carlo simulations. The user can either use the argument simu=TRUE to perform the Monte-Carlo simulation (with N the number of replications) or simply use the available tables by choosing simu=FALSE. In the latter case, the exact distribution is estimated for n=1, ...,150. For $151 <= n <= 175$, the distribution with n=150 is used. For $176 <= n <= 250$, the distribution with n=200 is used. For $251 <= n <= 400$, the distribution with n=300 is used. For $401 <= n <= 750$, the distribution with n=500 is used. For $n >= 751$, the distribution with n=1000 is used. Those tables were computed using 2e^5 replications in Monte-Carlo simulations.

Value

Returns the result of the test with its corresponding p-value and the value of the test statistic.

Note

Only a two sided alternative is possible with this test. Missing values are removed such that if a value of `x` (resp. `y`) is missing then the corresponding values of both `x` and `y` are removed. The test is then implemented on the remaining elements. If `ties.break="none"` the ties are ignored, putting mass (number of ties)/n at tied observations in the computation of the empirical cumulative distribution functions. If `ties.break="random"` they are randomly broken. If `ties.break="rep_random"` they are randomly broken `nb_tiebreak` times where `nb_tiebreak` is a parameter of the function. In that case, the test statistic and the p values are computed by taking the average over all replications.

This function is implemented using the Rcpp package.

Author(s)

See *Distribution Free Tests of Independence Based on the Sample Distribution Function*. J. R. Blum, J. Kiefer and M. Rosenblatt, 1961.

See Also

`cortest`, `vartest`, `mediantest`, `wilcoxtest`. See also the `hoeffd` function in the `Hmisc` package for the Hoeffding test.

Examples

```
#Simulated data 1
x<-c(0.2, 0.3, 0.1, 0.4)
y<-c(0.5, 0.4, 0.05, 0.2)
indeptest(x,y)

#Simulated data 2
n<-40 #sample size
x<-rnorm(n)
y<-x^2+0.3*rnorm(n)
plot(x,y)
indeptest(x,y)

#Application on the Evans dataset
```

```
#Description of this dataset is available in the lbreg package
data(Evans)
with(Evans,plot(CHL[CDH==1],DBP[CDH==1]))
with(Evans,cor.test(CHL[CDH==1],DBP[CDH==1])) #the standard Pearson test
with(Evans,cortest(CHL[CDH==1],DBP[CDH==1])) #the robust Pearson test
with(Evans,indeptest(CHL[CDH==1],DBP[CDH==1])) #the robust independence test
#The robust tests give very different pvalues than the standard Pearson test!

#Breaking the ties
#The ties are broken once
with(Evans,indeptest(CHL[CDH==1],DBP[CDH==1],ties.break="random"))
#The ties are broken repeatedly and the average of the test statistics and p.values
#are computed
with(Evans,indeptest(CHL[CDH==1],DBP[CDH==1],ties.break="rep_random",nb_tiebreak=100))
```

`mediantest` *Test for the median in the one and two-sample paired tests*

Description

In the one sample case, test if the median of the random variable is equal to 0. In the paired two-sample case, test if the median of the difference between the two random variables is equal to 0. The function also returns the confidence interval for the median of the random variable in the one-sample case or for the median of the difference between the two random variables in the two-sample case.

Usage

```
mediantest(
  x,
  y = NULL,
  alternative = "two.sided",
  paired = FALSE,
  conf.level = 0.95
)
```

Arguments

`x, y`	two continuous variables.
`alternative`	indicates the alternative hypothesis and must be one of "two.sided", "greater" or "less".
`paired`	a logical value. If `paired=TRUE`, the user must provide values for `x` and `y` and the paired test is implemented. If `paired=FALSE`, only `x` must be provided.
`conf.level`	confidence level for the confidence interval of the median.

Details

The null hypothesis for the one sample median test is: H0 Med(X)=0 where Med represents the median. The alternative is specified by the `alternative` argument: "`greater`" means that Med(X)>0, "`less`" means that Med(X)<0 and "`two.sided`" means that Med(X) is different from 0. The null hypothesis for the paired median test is: H0 Med(X-Y)=0. Both tests are asymptotically calibrated in the sense that the rejection probability under the null hypothesis is asymptotically equal to the level of the test. The test and the confidence interval are constructed based on asymptotic results on the rank statistics for the uniform distribution, as described in the book *Asymptotic Statistics* from A. W. Van der Vaart, 1998.

Value

Returns the result of the test with its corresponding p-value and the value of median of X or of X-Y. The confidence interval is also displayed but only when `alternative=two.sided`.

Note

The paired median test can be implemented either by providing the variables `x` and `y` with `paired=TRUE` or by just providing one vector equal to the difference between `x` and `y` with `paired=FALSE`.

Author(s)

See *Asymptotic Statistics*. A. W. Van der Vaart, 1998.

See Also

`cortest`, `indeptest`, `vartest`, `wilcoxtest`.

Examples

```
#Simulations
x <- rexp(200)-log(2)
mediantest(x)
x <- rexp(200)-log(2)+0.2
mediantest(x,alternative="greater")

n=100 #sample size
M=1000 #number of replications
res1=res2=rep(NA,M)
testone=function(n){
D=rchisq(n,df=4)-qchisq(df=4, p=0.5)
list(test1=mediantest(D)$p.value,test2=binom.test(sum(D>0),n)$p.value)
} #test2 is the sign test.
for (i in 1:M)
{
result=testone(n)
res1[i]=result$test1
res2[i]=result$test2
}
mean(res1<0.05) #0.048
mean(res2<0.05) # 0.04
```

`simulecdf` *Simulate the distribution of the test statistic for the robust independence test of Kolmogorov-Smirnov's type*

Description

For two independent continuous uniform variables on $[0, 1]$ compute the maximal distance between the joint empirical cumulative distribution function and the product of the marginal empirical cumulative distribution functions using Monte-Carlo simulations.

Usage

```
simulecdf(n, N)
```

Arguments

`n`	the size of the sample.
`N`	the number of replications in the Monte-Carlo simulation.

Details

Let $(x1, y1), ..., (x_n, y_n)$ be a bivariate sample of `n` independent continuous uniform variables. Its corresponding bivariate e.c.d.f. (empirical cumulative distribution function) Fn is defined as:

Fn(t1,t2) = `#{xi<=t1,yi<=t2}/n` = `sum_{i=1}^n Indicator(xi<=t1,yi<=t2)/n`.

Let Fn(t1) and Fn(t2) be the marginals e.c.d.f. Based on N Monte_Carlo simulations, the function computes the e.c.d.f. of

$$n^(1/2)sup_{t1,t2}|Fn(t1, t2) - Fn(t1) * Fn(t2)|.$$

Value

Returns the e.c.d.f. based on the N Monte_Carlo simulations. The returned object is a stepfun object obtained from the function `ecdf`.

See Also

`indeptest`, `stat_indeptest`, `ecdf2D`.

stat_indeptest *Compute the test statistic for the robust independence test of Kolmogorov-Smirnov's type*

Description

For two continuous variables compute the maximal distance between the joint empirical cumulative distribution function and the product of the marginal empirical cumulative distribution functions.

Usage

```
stat_indeptest(x, y)
```

Arguments

x, y	the two continuous variables. Must be of same length.

Details

Let (x1,y1), ..., (x_n,y_n) be a bivariate sample of n continuous variables. Its corresponding bivariate e.c.d.f. (empirical cumulative distribution function) Fn is defined as:

Fn(t1,t2) = `#{xi<=t1,yi<=t2}/n` = `sum_{i=1}^n Indicator(xi<=t1,yi<=t2)/n`.

Let Fn(t1) and Fn(t2) be the marginals e.c.d.f. The function returns the value of:

`n^(1/2) sup_{t1,t2} |Fn(t1,t2)-Fn(t1)*Fn(t2)|`.

Value

Returns the test statistic of the robust independent test.

See Also

indeptest, simulecdf, ecdf2D.

Examples

```
#Simulated data 1
x<-c(0.2, 0.3, 0.1, 0.4)
y<-c(0.5, 0.4, 0.05, 0.2)
stat_indeptest(x,y)

#Simulated data 2
n<-40
x<-rnorm(n)
y<-x^2+0.3*rnorm(n)
plot(x,y)
stat_indeptest(x,y)

#Application on the Evans dataset
```

```
data(Evans)
with(Evans,stat_indeptest(CHL[CDH==1],DBP[CDH==1]))
```

`tiebreak` *Break the ties in a given vector or between two vectors*

Description

If the vector contains ties (either inside a single or between two vectors), the function breaks them using a random perturbation.

Usage

```
tiebreak(x, y = NULL, nb_break = FALSE)
```

Arguments

x, y	the variables containing ties.
nb_break	if TRUE return also the number of values that have been broken

Details

If y=NULL the function detects the ties in the vector x. A uniform variable on the interval $[-e^(-5), e^(-5)]$ is added to the value of all the ties but one in the vector x. If y is also provided, the function detects the ties between x and y and break them (only in the x vector) by adding a uniform variable on the interval $[-e^(-5), e^(-5)]$ to these values. If nb_break=TRUE the result is returned as a list that also includes the number of values that have been broken.

Value

After breaking the ties, either returns the x variable as a vector or returns a list containing the x and y variables. If nb_break=TRUE, the number of values that have been broken is also an element of the list.

Examples

```
x <- c(1,2,2,3,4,5,5,5,7)
xbreak=tiebreak(x)
xbreak
#a uniform value has been added to the second, sixth and seventh value of x.
tiebreak(x,nb_break=TRUE) #3 values have been broken in x
sum(duplicated(xbreak))#check if the breaking procedure has worked.
y <- c(4,9,12,11,2,10)
xy_break=tiebreak(x,y)
xy_break$x
xy_break$y #a uniform value has been added to the second, third and fifth value of x.
xy_break$x%in%xy_break$y #check that no values for xbreak can be found in ybreak
tiebreak(x,y,nb_break=TRUE) #also returns the number of broken values
```

`vartest` *Calibrated variance test between two or more independent samples*

Description

Tests the equality of variance for continuous independent samples using Welch's ANOVA on the square of the centered variables.

Usage

```
vartest(x, y, ...)
```

Arguments

`x`, `y`	the two continuous variables for the two samples variance test.
`...`	other parameters such as `alternative` which indicates the alternative hypothesis as one of "two.sided", "greater" or "less", or `data` and `formula` to include a dataset with a formula from the command `vartest(x~y,data=dataset)` with dataset the name of the data.

Details

For the two sample variance test, the null hypothesis is: H0 var(X)=var(Y) where var represents the variance. The alternative is specified by the `alternative` argument. The test uses Welch's ANOVA procedure on the variables $(x_i - mean(x))^2$, $(y_i - mean(y))^2$ and is asymptotically well calibrated in the sense that the rejection probability under the null hypothesis is asymptotically equal to the level of the test. The test can be applied to more than two groups in which case it tests if all groups have the same variance.

Value

Returns the result of the test with its corresponding p-value and the value of the test statistic. In the two sample case, the function also returns the confidence interval for the difference of the two variances.

Note

The function can also be called using formulas: type `vartest(x~y,data)` with x the quantitative variable and y a factor variable with two or more levels. For more than two groups, only the formula is valid.

See Also

`cortest`, `indeptest`, `mediantest`, `wilcoxtest`.

Examples

```
#Application on the Evans dataset
data(Evans)
#Description of this dataset is available in the lbreg package
with(Evans,var.test(CHL[CDH==0],CHL[CDH==1]))
with(Evans,vartest(CHL[CDH==0],CHL[CDH==1]))
vartest(CHL~CDH,data=Evans) #using formulas

#Similar pvalues between var.test and vartest


#Simulated data
n=60 #sample size
M=10000 #number of replications
testone=function(n){
X=rnorm(n,0,1)
Y=rchisq(2*n,df=2)/2
list(test1=vartest(X,Y)$p.value,test2=var.test(X,Y)$p.value) #var.test is the standard Fisher test
}
res1=res2=rep(NA,M)
# Replications to check if the corrected Fisher test and the standard test are well calibrated
for (i in 1:M)
{
result=testone(n)
res1[i]=result$test1
res2[i]=result$test2
}
mean(res1<0.05)  #0.0515
mean(res2<0.05)  #0.1509
```

`wilcoxtest` *Calibrated Wilcoxon rank sum and signed rank tests*

Description

Compares the distribution between two random variables by testing if one variable tends to take larger (or smaller) values than the other. The test works for independent and paired variables by using corrected versions of the Wilcoxon (or equivalently Mann-Whitney in the two-sample case) for one and two-sample tests.

Usage

```
wilcoxtest(
  x,
  y = NULL,
  alternative = "two.sided",
  ties.break = "none",
  paired = FALSE,
```

```
  ...
)
```

Arguments

x, y	two continuous variables.
alternative	indicates the alternative hypothesis and must be one of "two.sided", "greater" or "less".
ties.break	the method used to break ties in case there are ties in the x or y vectors. Can be "none" or "random".
paired	a logical value. If paired=TRUE, you must provide values for x and y (of same length) and the paired test is implemented. If paired=FALSE, the paired test is implemented when y is null and only x is provided and the two sample test (for independent variables) is implemented when both x and y are provided.
...	it is possible to use a formula with or without specifying a dataset from the commands wilcoxtest(x~y) or wilcoxtest(x~y,data=dataset) with dataset the name of the data.

Details

For two independent samples, the null hypothesis for the corrected Wilcoxon (Mann-Whitney) test is: H0 Med(X-Y)=0 where Med represents the median. The alternative is specified by the alternative argument: "greater" means that Med(X-Y)>0 and "less" means that Med(X-Y)<0. The null hypothesis for the paired Wilcoxon test is: H0 Med(D1+D2)=0 where D1 is the difference between X1 and Y1 taken on the same pair (same with D2 on a different pair). Both tests are asymptotically well calibrated in the sense that the rejection probability under the null hypothesis is asymptotically equal to the level of the test.

Value

Returns the result of the test with its corresponding p-value and the value of the test statistic.

Note

The function can also be called using formulas: type wilcoxtest(x~y,data) with x the quantitative variable and y a factor variable with two levels. The option ties.break handles ties in the Wilcoxon test. If ties.break="none" the ties are ignored, if ties.break="random" they are randomly broken. For the Wilcoxon rank sum test the ties between the x and y are detected and broken (but the ties inside the x and y vectors are not changed). For the signed rank test, the ties in the vector x-y (or in the x vector in case y=NULL) are randomly broken.

See Also

cortest, indeptest, mediantest, vartest.

Examples

```
#Application on the Evans dataset
data(Evans)
#Description of this dataset is available in the lbreg package
with(Evans,wilcox.test(CHL[CDH==0],CHL[CDH==1]))
with(Evans,wilcoxtest(CHL[CDH==0],CHL[CDH==1]))
wilcoxtest(CHL~CDH,data=Evans) #using formulas
wilcoxtest(CHL~CDH,data=Evans,ties.break="random")
#the same test where ties are randomly broken


#For independent samples
n=100 #sample size
M=1000 #number of replications
testone=function(n){
X=runif(n,-0.5,0.5)
Y=rnorm(3*n,0,0.04)
list(test1=wilcoxtest(X,Y)$p.value,test2=wilcox.test(X,Y)$p.value)
#wilcox.test is the standard Wilcoxon test
}

#Simulation under the null hypothesis
#(note that P(X>Y)=0.5)
#Takes a few seconds to run
res1=res2=rep(NA,M)
for (i in 1:M)
{
result=testone(n)
res1[i]=result$test1
res2[i]=result$test2
}
mean(res1<0.05)
mean(res2<0.05)

#For paired samples
#We use the value of the median of a Gamma distributed variable with shape
#parameter equal to 1/5 and scale parameter equal to 1. This value is
#computed from the command qgamma(shape=1/5, scale=1, 0.5)
n=100 #sample size
M=1000 #number of replications
testone=function(n){
D=rgamma(n,shape=1/10,scale=1)-qgamma(shape=1/5, scale=1, 0.5)/2
list(test1=wilcoxtest(D,ties.break = "random")$p.value,test2=wilcox.test(D)$p.value)
#wilcox.test is the standard paired Wilcoxon test
}
#Simulation under the null hypothesis
#(note that Med(D_1+D_2)=0)
#Takes a few seconds to run
for (i in 1:M)
{
result=testone(n)
res1[i]=result$test1
```

```
res2[i]=result$test2
}
mean(res1<0.05)
mean(res2<0.05)
```

Index

∗ **datasets**
Evans, 6
∗ **test**
cortest, 2
indeptest, 7
mediantest, 9
vartest, 14
wilcoxtest, 15

cortest, 2, *8*, *10*, *14*, *16*

ecdf2D, 5, *11, 12*
Evans, 6

indeptest, *3*, *5*, 7, *10–12*, *14*, *16*

mediantest, *3*, *8*, 9, *14*, *16*

simulecdf, 11, *12*
stat_indeptest, *11*, 12

tiebreak, *3*, 13

vartest, *3*, *8*, *10*, 14, *16*

wilcoxtest, *3*, *8*, *10*, *14*, 15