

Package ‘independence’

August 24, 2020

Type Package

Title Fast Rank-Based Independence Testing

Version 1.0

Date 2020-08-10

Maintainer Chaim Even-Zohar <chaim@ucdavis.edu>

Description Performs three ranking-based nonparametric tests for the independence of two continuous variables:

- (1) the classical Hoeffding's D test;
- (2) a refined variant of it, named R;
- (3) the Bergsma-Dassios T* sign covariance.

The first test is consistent assuming an absolutely continuous bivariate distribution, i.e., the population coefficient $D=0$ iff the variables are independent.

The latter two are consistent under no restriction on the distribution.

All three statistics are computed in time $O(n \log n)$ given n iid paired samples.

The computation of R and T* uses a new algorithm, following work of Even-Zohar and Leng (2019), see <arXiv:1911.01414> and references therein.

License GPL (≥ 3)

Imports Rcpp ($\geq 1.0.5$)

LinkingTo Rcpp

Suggests TauStar, testthat

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation yes

Author Chaim Even-Zohar [aut, cre]

Repository CRAN

Date/Publication 2020-08-24 15:50:11 UTC

R topics documented:

.calc.hoeffding	2
.calc.refined	3
.calc.taustar	4
.generic.rank.test	5
hoeffding.D.test	7
hoeffding.refined.test	9
independence	11
max_hoeffding	12
max_taustar	13
relative.order	13
tau.star.test	14
Index	17

.calc.hoeffding	<i>Compute Hoeffding's D statistic</i>
-----------------	--

Description

This is an internal CPP function, used by the R function [hoeffding.D.test](#).

Usage

.calc.hoeffding(perm)

Arguments

perm	An integer vector containing exactly 0,1,...,n-1 in any order. The validity of the input is not checked by this function.
------	--

Details

Given (X1,Y1),...,(Xn,Yn), Hoeffding's Dn only depends on the permutation P that satisfies rank Yi = P[rank Xi]. This function computes Dn given P in O(n log n) time.

Value

Hoeffding's D statistic of perm.
The normalization is such that $-1/60 \leq D \leq 1/30$.
The return value -1.0 indicates an error.

Examples

```
.calc.hoeffding(0:4)
## [1] 0.03333333

.calc.hoeffding(c(0,3,2,1,4))
## [1] -0.01666667

set.seed(397)
.calc.hoeffding(order(runif(1000))-1) * 36
## [1] 0.004349087
```

<code>.calc.refined</code>	<i>Compute the refined Hoeffding statistic</i>
----------------------------	--

Description

This is an internal CPP function, used by the R function `hoeffding.refined.test`.

Usage

```
.calc.refined(perm)
```

Arguments

<code>perm</code>	An integer vector containing exactly 0,1,...,n-1 in any order. The validity of the input is not checked by this function.
-------------------	--

Details

Given $(X_1, Y_1), \dots, (X_n, Y_n)$, the refined Hoeffding statistic R_n only depends on the permutation P that satisfies $\text{rank } Y_i = P[\text{rank } X_i]$. This function computes R_n given P in $O(n \log n)$ time.

Value

The refined Hoeffding statistic of `perm`.

The normalization is such that $-1/180 \leq R \leq 1/90$.

The return value -1.0 indicates an error.

Examples

```
.calc.refined(0:4)
## [1] 0.01111111

.calc.refined(c(0,3,2,1,4))
## [1] -0.005555556
```

```
set.seed(397)
.calc.refined(order(runif(1000))-1) * 36
## [1] 0.004414034
```

<code>.calc.taustar</code>	<i>Compute the Tau^* statistic</i>
----------------------------	---

Description

This is an internal CPP function, used by the R function `tau.star.test`.

Usage

```
.calc.taustar(perm)
```

Arguments

<code>perm</code>	An integer vector containing exactly 0,1,...,n-1 in any order. The validity of the input is not checked by this function.
-------------------	--

Details

Given $(X_1, Y_1), \dots, (X_n, Y_n)$, the Tau^*_n statistic only depends on the permutation P that satisfies $\text{rank } Y_i = P[\text{rank } X_i]$. This function computes Tau^*_n given P in $O(n \log n)$ time.

Value

The Tau^* statistic of `perm`.

The normalization is such that $-1/3 \leq Tau^* \leq 2/3$.

The return value -1.0 indicates an error.

Examples

```
.calc.taustar(0:3)
## [1] 0.6666667

.calc.taustar(c(0,2,1,3))
## [1] -0.3333333

set.seed(397)
.calc.taustar(order(runif(1000))-1)
## [1] 0.004392385
```

<code>.generic.rank.test</code>	<i>Generic rank test for paired samples</i>
---------------------------------	---

Description

An internal function unifying several nonparametric tests for paired samples.

Usage

```
.generic.rank.test(  
  xs,  
  ys,  
  test,  
  letter,  
  description,  
  na.rm = TRUE,  
  collisions = TRUE,  
  precision = 1e-05,  
  limit_law_coef = 1,  
  min_samples = 1,  
  max_samples = Inf  
)
```

Arguments

<code>xs, ys</code>	Same-length numeric vectors, containing paired samples.
<code>test</code>	Function computing the test statistic given a relative order.
<code>letter</code>	Notation for the test statistic, e.g., "D" for Hoeffding's D.
<code>description</code>	Full name of test.
<code>na.rm</code>	Logical: Should missing values, NaN, and Inf be removed?
<code>collisions</code>	Logical: Warn of repeating values in <code>xs</code> or <code>ys</code> .
<code>precision</code>	of p-value, between 0 and 1. Otherwise p-value=NA.
<code>limit_law_coef</code>	Scaling of test statistic for standard null distribution.
<code>min_samples, max_samples</code>	Data size limits.

Details

The function `.generic.rank.test` first calls `relative.ordering` with `xs` and `ys`. Then it uses the given function to compute the test statistic from the resulting permutation. The statistic is rescaled by multiplication with $(n-1)*limit_law_coef$, where n is the sample size. Finally, it computes the p-value by calling `pHoeffInd` from the package `TauStar`.

Value

A list, of class "indtest":

method	the test's name
n	number of data points used
Tn/Dn/Rn/...	the test statistic, measure of dependence
scaled	the test statistic rescaled for a standard null distribution
p.value	the asymptotic p-value, by <code>TauStar::pHoeffInd</code>

P-value

The null distribution of the test statistic was described by Hoeffding. The p-value is approximated by calling the function `pHoeffInd` from the package `TauStar` by Luca Weihs.

By default, the p-value's precision parameter is set to $1e-5$. It seems that better precision would cost a considerable amount of time, especially for large values of the test statistic. It is therefore recommended to modify this parameter only upon need.

In case that `TauStar` is unavailable, or to save time in repeated use, set `precision = 1` to avoid computing p-values altogether. The scaled test statistic may be used instead. Its asymptotic distribution does not depend on any parameter. Also the raw test statistic may be used, descriptively, as a measure of dependence. Only its accuracy depends on the sample size.

Ties

This package currently assumes that the variables under consideration are non-atomic, so that ties are not expected, other than by occasional effects of numerical precision. Addressing ties rigorously is left for future versions.

The flag `collisions = TRUE` invokes checking for ties in `xs` and in `ys`, and produces an appropriate warning if they exist. The current implementation breaks such ties arbitrarily, not randomly.

By the averaging nature of the test statistic, it seems that a handful of ties should not be of much concern. In case of more than a handful of ties, our current advice to the user is to break them uniformly at random beforehand.

Big Data

The test statistic is computed in almost linear time, $O(n \log n)$, given a sample of size n . Its computation involves integer arithmetics of order n^4 or n^5 , which should fit into an integer data type supported by the compiler.

Most 64-bit compilers emulate 128-bit arithmetics. Otherwise we use the standard 64-bit arithmetics. Find the upper limits of your environment using

1. `max_taustar()`
2. `max_hoeffding()`

Another limitation is $2^{31}-1$, the maximum size and value of an integer vector in a 32-bit build of R. This is only relevant for the tau star statistic in 128-bit mode, which could otherwise afford about three times that size. If your sample size falls in this range, try recompiling the function `.calc.taustar` according to the instructions in the cpp source file.

See Also

`independence`, `relative.order`, `tau.star.test`, `hoeffding.D.test`, `hoeffding.refined.test`

hoeffding.D.test	<i>Hoeffding's D independence test</i>
------------------	--

Description

The function `hoeffding.D.test` provides independence testing for two continuous numeric variables, that is consistent for absolutely-continuous alternative bivariate distributions. It implements the classical D statistic by Hoeffding, which in terms of CDFs estimates the integral of $(F_{xy} - F_x F_y)^2 dF_{xy}$. It may also be expressed in terms of the ordering types of quintuples of data points. Its efficient $O(n \log n)$ computation seems to be new in R.

Usage

```
hoeffding.D.test(xs, ys, na.rm = TRUE, collisions = TRUE, precision = 1e-05)
```

Arguments

<code>xs, ys</code>	Same-length numeric vectors, containing paired samples.
<code>na.rm</code>	Logical: Should missing values, NaN, and Inf be removed?
<code>collisions</code>	Logical: Warn of repeating values in <code>xs</code> or <code>ys</code> .
<code>precision</code>	of p-value, between 0 and 1. Otherwise p-value=NA.

Value

A list, of class "indtest":

<code>method</code>	the test's name
<code>n</code>	number of data points used
<code>Tn/Dn/Rn/...</code>	the test statistic, measure of dependence
<code>scaled</code>	the test statistic rescaled for a standard null distribution
<code>p.value</code>	the asymptotic p-value, by <code>TauStar::pHoeffInd</code>

P-value

The null distribution of the test statistic was described by Hoeffding. The p-value is approximated by calling the function `pHoeffInd` from the package `TauStar` by Luca Weihs.

By default, the p-value's precision parameter is set to $1e-5$. It seems that better precision would cost a considerable amount of time, especially for large values of the test statistic. It is therefore recommended to modify this parameter only upon need.

In case that `TauStar` is unavailable, or to save time in repeated use, set `precision = 1` to avoid computing p-values altogether. The scaled test statistic may be used instead. Its asymptotic distribution does not depend on any parameter. Also the raw test statistic may be used, descriptively, as a measure of dependence. Only its accuracy depends on the sample size.

Ties

This package currently assumes that the variables under consideration are non-atomic, so that ties are not expected, other than by occasional effects of numerical precision. Addressing ties rigorously is left for future versions.

The flag `collisions = TRUE` invokes checking for ties in `xs` and in `ys`, and produces an appropriate warning if they exist. The current implementation breaks such ties arbitrarily, not randomly.

By the averaging nature of the test statistic, it seems that a handful of ties should not be of much concern. In case of more than a handful of ties, our current advice to the user is to break them uniformly at random beforehand.

Big Data

The test statistic is computed in almost linear time, $O(n \log n)$, given a sample of size n . Its computation involves integer arithmetics of order n^4 or n^5 , which should fit into an integer data type supported by the compiler.

Most 64-bit compilers emulate 128-bit arithmetics. Otherwise we use the standard 64-bit arithmetics. Find the upper limits of your environment using

1. `max_tau_star()`
2. `max_hoeffding()`

Another limitation is $2^{31}-1$, the maximum size and value of an integer vector in a 32-bit build of R. This is only relevant for the tau star statistic in 128-bit mode, which could otherwise afford about three times that size. If your sample size falls in this range, try recompiling the function `.calc.tau_star` according to the instructions in the cpp source file.

References

Hoeffding, Wassily. "A non-parametric test of independence." The annals of mathematical statistics (1948): 546-557.

Luca Weihs (2019). TauStar: Efficient Computation and Testing of the Bergsma-Dassios Sign Covariance. R package version 1.1.4. <https://CRAN.R-project.org/package=TauStar>

Frank E Harrell Jr, with contributions from Charles Dupont and many others. (2020). Hmisc: Harrell Miscellaneous. R package version 4.4-0. <https://CRAN.R-project.org/package=Hmisc>

See Also

[independence](#), [tau.star.test](#), [hoeffding.refined.test](#)

Examples

```
## independent, $p.value is 0.2582363
set.seed(123)
hoeffding.D.test(rnorm(10000),rnorm(10000))

## dependent, even though uncorrelated, $p.value is 0.0002891223
set.seed(123)
```

```
hoeffding.D.test(rnorm(10000,0,3001:13000), rnorm(10000,0,3001:13000))
```

hoeffding.refined.test

The refined Hoeffding independence test

Description

The function `hoeffding.refined.test` provides independence testing for two continuous numeric variables, that is consistent for all alternatives. The test statistic is a variant of the classical Hoeffding's D statistic. In terms of CDFs, it estimates the integral of $(F_{xy} - F_x * F_y)^2 dF_x dF_y$, based on the ordering types of quintuples of data points. This test statistic is efficiently computed via a new $O(n \log n)$ -time algorithm, following work of Even-Zohar and Leng.

Usage

```
hoeffding.refined.test(
  xs,
  ys,
  na.rm = TRUE,
  collisions = TRUE,
  precision = 1e-05
)
```

Arguments

<code>xs, ys</code>	Same-length numeric vectors, containing paired samples.
<code>na.rm</code>	Logical: Should missing values, NaN, and Inf be removed?
<code>collisions</code>	Logical: Warn of repeating values in <code>xs</code> or <code>ys</code> .
<code>precision</code>	of p-value, between 0 and 1. Otherwise p-value=NA.

Value

A list, of class "indtest":

<code>method</code>	the test's name
<code>n</code>	number of data points used
<code>Tn/Dn/Rn/...</code>	the test statistic, measure of dependence
<code>scaled</code>	the test statistic rescaled for a standard null distribution
<code>p.value</code>	the asymptotic p-value, by <code>TauStar::pHoeffInd</code>

P-value

The null distribution of the test statistic was described by Hoeffding. The p-value is approximated by calling the function `pHoeffInd` from the package `TauStar` by Luca Weihs.

By default, the p-value's precision parameter is set to $1e-5$. It seems that better precision would cost a considerable amount of time, especially for large values of the test statistic. It is therefore recommended to modify this parameter only upon need.

In case that `TauStar` is unavailable, or to save time in repeated use, set `precision = 1` to avoid computing p-values altogether. The scaled test statistic may be used instead. Its asymptotic distribution does not depend on any parameter. Also the raw test statistic may be used, descriptively, as a measure of dependence. Only its accuracy depends on the sample size.

Ties

This package currently assumes that the variables under consideration are non-atomic, so that ties are not expected, other than by occasional effects of numerical precision. Addressing ties rigorously is left for future versions.

The flag `collisions = TRUE` invokes checking for ties in `xs` and in `ys`, and produces an appropriate warning if they exist. The current implementation breaks such ties arbitrarily, not randomly.

By the averaging nature of the test statistic, it seems that a handful of ties should not be of much concern. In case of more than a handful of ties, our current advice to the user is to break them uniformly at random beforehand.

Big Data

The test statistic is computed in almost linear time, $O(n \log n)$, given a sample of size n . Its computation involves integer arithmetics of order n^4 or n^5 , which should fit into an integer data type supported by the compiler.

Most 64-bit compilers emulate 128-bit arithmetics. Otherwise we use the standard 64-bit arithmetics. Find the upper limits of your environment using

1. `max_taustar()`
2. `max_hoeffding()`

Another limitation is $2^{31}-1$, the maximum size and value of an integer vector in a 32-bit build of R. This is only relevant for the tau star statistic in 128-bit mode, which could otherwise afford about three times that size. If your sample size falls in this range, try recompiling the function `.calc.taustar` according to the instructions in the cpp source file.

References

Hoeffding, Wassily. "A non-parametric test of independence." The annals of mathematical statistics (1948): 546-557.

Yanagimoto, Takemi. "On measures of association and a related problem." Annals of the Institute of Statistical Mathematics 22.1 (1970): 57-63.

Luca Weihs (2019). `TauStar`: Efficient Computation and Testing of the Bergsma-Dassios Sign Covariance. R package version 1.1.4. <https://CRAN.R-project.org/package=TauStar>

Even-Zohar, Chaim. "Patterns in Random Permutations." arXiv preprint arXiv:1811.07883 (2018).

Even-Zohar, Chaim, and Calvin Leng. "Counting Small Permutation Patterns." arXiv preprint arXiv:1911.01414 (2019).

See Also

[independence](#), [tau.star.test](#), [hoeffding.D.test](#),

Examples

```
## independent, $p.value is 0.258636
set.seed(123)
hoeffding.refined.test(rnorm(10000),rnorm(10000))

## dependent, even though uncorrelated, $p.value is 0.0003017679
set.seed(123)
hoeffding.refined.test(rnorm(10000,0,3001:13000), rnorm(10000,0,3001:13000))
```

independence

independence

Description

Fast Rank-Based Independence Testing

Details

The package `independence` provides three ranking-based nonparametric tests for the independence of two continuous variables X and Y :

1. the classical Hoeffding's D test: [hoeffding.D.test](#)
2. a refined variant of it, named R : [hoeffding.refined.test](#)
3. the Bergsma-Dassios T^* sign covariance: [tau.star.test](#)

The first test is consistent assuming an absolutely continuous joint distribution, i.e., the population coefficient $D=0$ iff the variables are independent. The latter two are consistent under no restriction on the distribution.

Given an iid sample $(X_1, Y_1), \dots, (X_n, Y_n)$, all three statistics are computed in time $O(n \log n)$ improving upon previous implementations. The statistics R and T^* are computed by a new algorithm, following work of Even-Zohar and Leng. It is based on the fast counting of certain patterns in the permutation that relates the ranks of X and Y . See [arxiv:1911.01414] and references therein.

Author(s)

Chaim Even-Zohar <<chaim@ucdavis.edu>>

See Also

[tau.star.test](#), [hoeffding.D.test](#), [hoeffding.refined.test](#) [relative.order](#)

Examples

```
library(independence)

## independent
set.seed(123)
xs = rnorm(10000)
ys = rnorm(10000)
hoeffding.D.test(xs,ys)
hoeffding.refined.test(xs,ys)
tau.star.test(xs,ys)

## dependent, even though uncorrelated
set.seed(123)
xs = rnorm(10000,0,3001:13000)
ys = rnorm(10000,0,3001:13000)
hoeffding.D.test(xs,ys)
hoeffding.refined.test(xs,ys)
tau.star.test(xs,ys)

## dependent but not absolutely continuous, fools Hoeffding's D
set.seed(123)
xs = runif(200)
f = function(x,y) ifelse(x>y, pmin(y,x/2), pmax(y,(x+1)/2))
ys = f(xs,runif(200))
hoeffding.D.test(xs,ys)
hoeffding.refined.test(xs,ys)
tau.star.test(xs,ys)
```

max_hoeffding

Maximum data size for Hoeffding's statistics

Description

Data at most this size should not cause an overflow in the computation of Hoeffding's D statistic or its refinement. This may depend on the availability of 64-bit or 128-bit words to the compiler in use.

Usage

```
max_hoeffding()
```

Value

100413509 in 128-bit mode, 14081 in 64-bit mode.

See Also

[hoeffding.D.test](#) [hoeffding.refined.test](#)

Examples

```
max_hoeffding()
```

max_tastar

Maximum data size for the T^ statistic*

Description

Data at most this size should not cause an overflow in the computation of Tau^* . This may depend on the availability of 64-bit or 128-bit words to the compiler in use.

Usage

```
max_tastar()
```

Value

6721987087 in 128-bit mode, 102569 in 64-bit mode.

See Also

[tau.star.test](#)

Examples

```
max_tastar()
```

relative.order

The relative order of two vectors

Description

Given $(X_1, Y_1), \dots, (X_n, Y_n)$, many nonparametric statistics depend only on the permutation P that satisfies $\text{rank } Y_i = P[\text{rank } X_i]$. The function `relative_order` computes such P given X_1, \dots, X_n and Y_1, \dots, Y_n .

Usage

```
relative.order(xs, ys, na.rm = TRUE, collisions = TRUE)
```

Arguments

<code>xs, ys</code>	Numeric vectors of same length.
<code>na.rm</code>	Logical: Should missing values, <code>NaN</code> , and <code>Inf</code> be removed?
<code>collisions</code>	Logical: Warn of repeating values in <code>xs</code> or <code>ys</code> .

Details

By default, the function removes missing values, and warns of repeating values. Then it computes the relative order by calling the base R function `order` twice: `order(xs[order(ys)])`.

Ties may be broken arbitrarily, depending on the behavior of the function `order`.

Value

An integer vector which describes the ordering of the second argument `ys`, in terms of the ordering of the corresponding values in the first argument `xs`.

For example, if `xs[3]` is the *i*th smallest and `ys[3]` is the *j*th smallest, then the returned value in position *i* is *j*.

Examples

```
relative.order(1:5, c(10,30,50,40,20))
## [1] 1 3 5 4 2

relative.order(c(1,2,5,3,4), c(10,30,50,40,20))
## [1] 1 3 4 2 5

set.seed(123)
relative.order(runif(8), runif(8))
## [1] 5 4 8 1 3 2 7 6
```

tau.star.test

The Bergsma–Dassios tau independence test*

Description

The function `tau.star.test` provides an independence test for two continuous numeric variables, that is consistent for all alternatives. It is based on the Yanagimoto-Bergsma-Dassios coefficient, which compares the frequencies of concordant and discordant quadruples of data points. The test statistic is efficiently computed in $O(n \log n)$ time, following work of Even-Zohar and Leng.

Usage

```
tau.star.test(xs, ys, na.rm = TRUE, collisions = TRUE, precision = 1e-05)
```

Arguments

<code>xs, ys</code>	Same-length numeric vectors, containing paired samples.
<code>na.rm</code>	Logical: Should missing values, NaN, and Inf be removed?
<code>collisions</code>	Logical: Warn of repeating values in <code>xs</code> or <code>ys</code> .
<code>precision</code>	of p-value, between 0 and 1. Otherwise p-value=NA.

Value

A list, of class "indtest":

method	the test's name
n	number of data points used
Tn/Dn/Rn/...	the test statistic, measure of dependence
scaled	the test statistic rescaled for a standard null distribution
p.value	the asymptotic p-value, by <code>TauStar::pHoeffInd</code>

P-value

The null distribution of the test statistic was described by Hoeffding. The p-value is approximated by calling the function `pHoeffInd` from the package `TauStar` by Luca Weihs.

By default, the p-value's precision parameter is set to $1e-5$. It seems that better precision would cost a considerable amount of time, especially for large values of the test statistic. It is therefore recommended to modify this parameter only upon need.

In case that `TauStar` is unavailable, or to save time in repeated use, set `precision = 1` to avoid computing p-values altogether. The scaled test statistic may be used instead. Its asymptotic distribution does not depend on any parameter. Also the raw test statistic may be used, descriptively, as a measure of dependence. Only its accuracy depends on the sample size.

Ties

This package currently assumes that the variables under consideration are non-atomic, so that ties are not expected, other than by occasional effects of numerical precision. Addressing ties rigorously is left for future versions.

The flag `collisions = TRUE` invokes checking for ties in `xs` and in `ys`, and produces an appropriate warning if they exist. The current implementation breaks such ties arbitrarily, not randomly.

By the averaging nature of the test statistic, it seems that a handful of ties should not be of much concern. In case of more than a handful of ties, our current advice to the user is to break them uniformly at random beforehand.

Big Data

The test statistic is computed in almost linear time, $O(n \log n)$, given a sample of size n . Its computation involves integer arithmetics of order n^4 or n^5 , which should fit into an integer data type supported by the compiler.

Most 64-bit compilers emulate 128-bit arithmetics. Otherwise we use the standard 64-bit arithmetics. Find the upper limits of your environment using

1. `max_taustar()`
2. `max_hoeffding()`

Another limitation is $2^{31}-1$, the maximum size and value of an integer vector in a 32-bit build of R. This is only relevant for the tau star statistic in 128-bit mode, which could otherwise afford about three times that size. If your sample size falls in this range, try recompiling the function `.calc.taustar` according to the instructions in the cpp source file.

References

Bergsma, Wicher; Dassios, Angelos. A consistent test of independence based on a sign covariance related to Kendall's tau. *Bernoulli* 20 (2014), no. 2, 1006–1028.

Yanagimoto, Takemi. "On measures of association and a related problem." *Annals of the Institute of Statistical Mathematics* 22.1 (1970): 57-63.

Luca Weihs (2019). TauStar: Efficient Computation and Testing of the Bergsma-Dassios Sign Covariance. R package version 1.1.4. <https://CRAN.R-project.org/package=TauStar>

Even-Zohar, Chaim, and Calvin Leng. "Counting Small Permutation Patterns." arXiv preprint arXiv:1911.01414 (2019).

See Also

[independence](#), [hoeffding.D.test](#), [hoeffding.refined.test](#)

Examples

```
## independent, $p.value is 0.2585027
set.seed(123)
tau.star.test(rnorm(10000), rnorm(10000))

## dependent, even though uncorrelated, $p.value is 0.000297492
set.seed(123)
tau.star.test(rnorm(10000, 0, 3001:13000), rnorm(10000, 0, 3001:13000))
```

Index

`.calc.hoeffding`, 2
`.calc.refined`, 3
`.calc.taustar`, 4
`.generic.rank.test`, 5
`_PACKAGE (independence)`, 11

`Bergsma-Dassios (tau.star.test)`, 14

`calc.taustar`, 6, 8, 10, 15
`calc.taustar (.calc.taustar)`, 4

`dependence-test (independence)`, 11

`Hoeffding-test (hoeffding.D.test)`, 7
`hoeffding.D.test`, 2, 6, 7, 11, 12, 16
`hoeffding.refined.test`, 3, 6, 8, 9, 11, 12, 16

`independence`, 6, 8, 11, 11, 16
`independence-test (independence)`, 11

`max_hoeffding`, 6, 8, 10, 12, 15
`max_taustar`, 6, 8, 10, 13, 15

`order`, 14

`pHoeffInd`, 5–7, 9, 15

`refined-Hoeffding-test (hoeffding.refined.test)`, 9
`relative.order`, 6, 11, 13

`tau.star.test`, 4, 6, 8, 11, 13, 14
`taustar (tau.star.test)`, 14

`Yanagimoto-Bergsma-Dassios (tau.star.test)`, 14