Package 'multivariance'

January 10, 2018

Title Measuring Multivariate Dependence Using Distance Multivariance

Version 1.1.0

Date 2018-01-09

Description Distance multivariance is a measure of dependence which can be used to detect and quantify dependence. The necessary functions are implemented in this packages, and examples are given. For the theoretic background we refer to the papers:
B. Böttcher, M. Keller-Ressel, R.L. Schilling, Detecting independence of random vectors I. Generalized distance covariance and Gaussian covariance. Preprint 2017, <arXiv:1711.07778>.
B. Böttcher, M. Keller-Ressel, R.L. Schilling, Detecting independence of random vectors II. Distance multivariance and Gaussian multivariance. Preprint 2017, <arXiv:1711.07775>.
B. Böttcher, Dependence Structures -
Estimation and Visualization Using Distance Multivariance. Preprint 2017, <arXiv:1712.06532>.

Depends R (>= 3.3.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports abind, igraph, graphics, stats

RoxygenNote 6.0.1

Suggests testthat

NeedsCompilation no

Author Björn Böttcher [aut, cre],
Martin Keller-Ressel [ctb]

Maintainer Björn Böttcher <bjoern.boettcher@tu-dresden.de>

Repository CRAN

Date/Publication 2018-01-10 16:12:05 UTC

R topics documented:

`multivariance-package` *multivariance: Measuring Multivariate Dependence Using Distance Multivariance*

Description

The multivariance package provides basic functions to calculate distance multivariance and related quantities.

Details

Distance multivariance is a measure of dependence which can be used to detect and quantify dependence structures. The necessary functions are implemented in this packages, and examples are given. For the theoretic background we refer to the papers [1,2] and [3]. The latter includes a summary of the first two. It is the recommended starting point for users with an applied interest.

The (current) code should be understood as *proof of concept*, certainly there is room for improvement and development. Questions, comments and remarks are welcome: <bjoern.boettcher@tu-dresden.de>

For infos on the latest changes and/or updates to the package use `news(package="multivariance")`.

To cite this package use the standard citation for R packages, i.e., the output of `citation("multivariance")`.

Multivariance

`multivariance` computes the distance multivariance

`total.multivariance` computes the total distance multivariance

`m.multivariance` computes the m-multivariance (introduced in [3])

It might be convenient to compute these simultaneously using `multivariances.all`.

Functions to use and interpret multivariance

`rejection.level` computes the rejection level for a given significance level. This can be used for a conservative interpretation of distance multivariance. The counterpart is `multivariance.pvalue`, which computes a conservative p-value for a given distance multivariance. Both methods are distribution-free.

`resample.rejection.level` and `resample.pvalue` are the distribution dependent versions of the above. They are approximately sharp, but computational more expensive. Any resampling is done by `resample.multivariance`.

`independence.test` provides the corresponding tests of independence.

`cdm` and `cdms` compute the centered distance matrix and matrices, respectively. These can be used to speed up repeated computations of distance multivariance.

Dependence structures

`dependence.structure` performs the dependence structure detection algorithm as described in [3].

`find.cluster` is the basic building block of `dependence.structure`. It is recommended to use `dependence.structure`.

Examples

`coins` and `tetrahedron` generate samples of pairwise independent random variables, with dependence of higher order.

`dep_struct_iterated_13_100`, `dep_struct_ring_15_100`, `dep_struct_several_26_100` and `dep_struct_star_9_100` are example data sets for the dependence structure detection. These might also serve as benchmark examples.

References

[1] B. Böttcher, M. Keller-Ressel, R.L. Schilling, Detecting independence of random vectors I. Generalized distance covariance and Gaussian covariance. Preprint 2017. https://arxiv.org/abs/1711.07778

[2] B. Böttcher, M. Keller-Ressel, R.L. Schilling, Detecting independence of random vectors II. Distance multivariance and Gaussian multivariance. Preprint 2017. https://arxiv.org/abs/1711.07775

[3] B. Böttcher, Dependence Structures - Estimation and Visualization Using Distance Multivariance. Preprint 2017. https://arxiv.org/abs/1712.06532

cdm *centered distance matrix*

Description

computes the centered distance matrix

Usage

```
cdm(x, normalize = TRUE, psi = NULL, p = NULL)
```

Arguments

x	matrix, each row of the matrix is treated as one sample
normalize	logical, indicates if the matrix should be normalized
psi	a real valued function of two variables, to compute the distance of two samples based on a continuous negative definite function. If it is NULL, the euclidean distance will be used
p	numeric, if it is a value between 1 and 2 then the Minkowski distance with parameter p is used.

Details

The centered distance matrices are required for the computation of (total / m-) multivariance.

If `normalize = TRUE` then the value of multivariance is comparable and meaningful. It can be compared to the `rejection.level` or its p-value `multivariance.pvalue` can be computed.

More details: If `normalize = TRUE` the matrix is scaled such that the multivariance based on it, times the sample size, has in the limit - in the case of independence - the distribution of an L^2 norm of a Gaussian process with known expectation.

References

For the theoretic background see the references given on the main help page of this package: multivariance-package.

Examples

```
x = coins(100)
cdm(x) # fast euclidean distances
cdm(x,psi = function(x,y) sqrt(sum((x-y)^2))) # this is identical to the previous (but slower)

# the function cdm does the following three lines in a faster way
N = nrow(x)
C = diag(N) - matrix(1/N,nrow = N,ncol = N)
A = - C %*% as.matrix(stats::dist(x,method="euclidean")) %*% C #'
all(abs(A- cdm(x,normalize = FALSE)) < 10^(-12))
```

cdms *computes the centered distance matrices*

Description

computes the centered distance matrices

Usage

```
cdms(x, vec = 1:ncol(x), membership = NULL, ...)
```

Arguments

x	matrix, each row is a sample
vec	vector which indicates which columns are treated as one sample
membership	depreciated. Now use vec.
...	these are passed to cdm

Value

It returns an 3 dimensional array of the distance matrices. The index of the first dimension names the component for which the matrix is computed, thus it ranges from 1 to max(vec).

clean.graph *cleanup dependence structure graph*

Description

Given a dependence structure graph: vertices representing the multivariances of only two verticies become an edge labeled with the label of the vertex.

Usage

```
clean.graph(g)
```

Arguments

g	graph, created by dependence.structure

Value

graph

Examples

```
N = 200
y = coins(N,2)
x = cbind(y,y,y)
ds = dependence.structure(x)
plot(clean.graph(ds$graph))
```

`coins` *dependence example: k-independent coin sampling*

Description

This function creates samples which are dependent but k-independent.

Usage

```
coins(N = 1000, k = 2)
```

Arguments

`N`	number of samples
`k`	each k-tuple will be independent

Value

It returns the samples as rows of an `N` by `k+1` matrix. The columns are dependent but k-independent.

References

For the theoretic background see the reference [3] given on the main help page of this package: multivariance-package.

Examples

```
coins(200,4)
```

`dependence.structure` *determines the dependence structure*

Description

Determines the dependence structure as described in [3].

Usage

```
dependence.structure(x, vec = 1:ncol(x), verbose = TRUE,
  detection.aim = NULL, ...)
```

Arguments

`x`	matrix, each row of the matrix is treated as one sample
`vec`	vector, it indicates which columns are initially treated together as one sample
`verbose`	boolean, if `TRUE` details are printed during the detection and whenever a cluster is newly detected the (so far) detected dependence structure is plotted.
`detection.aim`	`=NULL` or a list of vectors which indicate the expected detection, see below for more details
`...`	these are passed to `find.cluster`

Details

Performs the detection of the dependence structure as described in [3].

If `fixed.rejection.level` is not provided, the significance level `alpha` is used to determine which multivariances are significant using the distribution-free rejection level. As default the Holm method is used for p-value correction corresponding to multiple testing.

The resulting graph can be simplified (pairwise dependence can be represented by edges instead of vertices) using `clean.graph`.

Advanced: The argument `detection.aim` can be used to check, if an expected dependence structure was detected. This might be useful for simulation studies to determine the empirical power of the detection algorithm. Hereto `detection.aim` is set to a list of vectors which indicate the expected detected dependence structures (one for each run of `find.cluster`). The vector has as first element the `k` for which k-tuples are detected (for this aim the detection stops without success if no k-tuple is found), and the other elements, indicate to which clusters all present vertices belong after the detection, e.g. `c(3,2,2,1,2,1,1,2,1)` expects that 3-tuples are detected and in the graph are 8 vertices (including those representing the detected 3 dependences), the order of the 2's and 1's indicate which vertices belong to which cluster. If `detection.aim` is provided, the vector representing the actual detection is printed, thus one can use the output with copy-paste to fix successively the expected detection aims.

Note that a failed detection might invoce the warning:

```
run$mem == detection.aim[[k]][-1] :
longer object length is not a multiple of shorter object length
```

Value

returns a list with elements:

`multivariances` calculated multivariances,

`cdms` calculated centered distance matrices,

`graph` graph representing the dependence structure.

`detected` boolean, this is only included if a `detection.aim` is given.

References

For the theoretic background see the reference [3] given on the main help page of this package: multivariance-package.

Examples

```
# structures for the datasets included in the package
dependence.structure(dep_struct_several_26_100)
dependence.structure(dep_struct_star_9_100)
dependence.structure(dep_struct_iterated_13_100)
dependence.structure(dep_struct_ring_15_100)

# basic examples:

dependence.structure(coins(100)) # 3-dependent
dependence.structure(coins(100),vec = c(1,1,2))
# 3-dependent rv of which the first two rv are used together as one rv, thus 2-dependence.

dependence.structure(cbind(coins(200),coins(200,k=5)),verbose = TRUE)
#1,2,3 are 3-dependent, 4,..,9 are 6-dependent

# similar to the the previous example, but
# the pair 1,2 is treated as one sample,
# anagously the pair 3,4. In the resulting structure one does not
# see anymore that the dependence of 1,2,3,4 with the rest is due
# to 4.
dependence.structure(cbind(coins(200),coins(200,k=5)),
                           vec = c(1,2,1,2,3,4,5,6,7),verbose = TRUE)


### Advanced:

# How to check the empirical power of the detection algorithm?
# Use a dataset for which the structure is detected, e.g. dep_struct_several_26_100.
# run:
dependence.structure(dep_struct_several_26_100,
                     detection.aim = list(c(ncol(dep_struct_several_26_100))))
# The output provides the first detection aim. Now we run the same line with the added
# detection aim
dependence.structure(dep_struct_several_26_100,detection.aim = list(c(3,1, 1, 1, 2, 2, 2, 3, 4,
  5, 6, 7, 8, 8, 8, 9, 9, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1, 2, 8, 9),
```

```
  c(ncol(dep_struct_several_26_100))))
# and get the next detection aim ... thus we finally obtain all detection aims.
# now we can run the code with new sample data ....
N = 100
dependence.structure(cbind(coins(N,2),tetrahedron(N),coins(N,4),tetrahedron(N),
                           tetrahedron(N),coins(N,3),coins(N,3),rnorm(N)),
                     detection.aim = list(c(3,1, 1, 1, 2, 2, 2, 3, 4, 5, 6, 7, 8, 8, 8,
  9, 9, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1, 2, 8, 9),
  c(4,1, 1, 1, 2, 2, 2, 3, 4, 5, 6, 7, 8, 8, 8, 9, 9, 9, 10, 10, 10, 10, 11, 11, 11,
    11, 12, 1, 2, 8, 9, 10, 11),
  c(5, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 8, 1,
    2, 4, 5, 6, 7, 3),
  c(5, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 8, 1,
    2, 4, 5, 6, 7, 3)))$detected
# ... and one could start to store the results and compute the rate of successes.

# ... or one could try to check how many samples are necessary for the detection:
re = numeric(100)
for (i in 2:100) {
  re[i] =
    dependence.structure(dep_struct_several_26_100[1:i,],verbose = FALSE,
                         detection.aim = list(c(3,1, 1, 1, 2, 2, 2, 3, 4, 5, 6, 7, 8,
      8, 8, 9, 9, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1, 2, 8, 9),
      c(4,1, 1, 1, 2, 2, 2, 3, 4, 5, 6, 7, 8, 8, 8, 9, 9, 9, 10, 10, 10, 10, 11, 11,
        11, 11, 12, 1, 2, 8, 9, 10, 11),
      c(5, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7,
        8, 1, 2, 4, 5, 6, 7, 3),
      c(5, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7,
        8, 1, 2, 4, 5, 6, 7, 3)))$detected
  print(paste("First", i,"samples. Detected?", re[i]==1))
}
cat(paste("Given the 1 to k'th row the structure is not detected for k =",which(re == FALSE),"\n"))
```

`dep_struct_iterated_13_100`

example dataset for `dependence.structure`

Description

It was generated by

```
set.seed(532333356)
N = 100
x = matrix(sample.int(2,10*N,replace = TRUE)-1,ncol = 10)
for (i in c(2,5,9)) x = cbind(x,(rowSums(as.matrix(x[,1:(i-1)])))
dep_struct_iterated_13_100 = x
save(dep_struct_iterated_13_100,file ="dep_struct_iterated_13_100.rda")
```

Usage

```
dep_struct_iterated_13_100
```

Format

`matrix` 13 variables (columns), 100 independent samples (rows)

Details

To avoid irritation, note that the seed is just a simple integer hash value of the variable name.

`dep_struct_ring_15_100`

example dataset for `dependence.structure`

Description

It was generated by

```
set.seed(436646700)
N = 100
n= 15
x=matrix(sample.int(2,N*n,replace = TRUE)-1,nrow =N)
x[,4] = rowSums(x[,1:3])
x[,7] = rowSums(x[,4:6])
x[,10] = rowSums(x[,7:9])
x[,13] = rowSums(x[,10:12])
x[,15] = rowSums(x[,c(13,14,1)])
dep_struct_ring_15_100 = x
save(dep_struct_ring_15_100,file ="dep_struct_ring_15_100.rda")
```

Usage

```
dep_struct_ring_15_100
```

Format

`matrix` 13 variables (columns), 100 independent samples (rows)

Details

To avoid irritation, note that the seed is just a simple integer hash value of the variable name.

---

dep_struct_several_26_100 *example dataset for* dependence.structure

---

Description

It was generated by

```
set.seed(1348879148)
N = 100
dep_struct_several_26_100 = cbind(coins(N,2),tetrahedron(N),coins(N,4),
    tetrahedron(N),tetrahedron(N),coins(N,3),coins(N,3),rnorm(N))
save(dep_struct_several_26_100,file ="dep_struct_several_26_100.rda")
```

Usage

```
dep_struct_several_26_100
```

Format

matrix 26 variables (columns), 100 independent samples (rows)

Details

To avoid irritation, note that the seed is just a simple integer hash value of the variable name.

---

dep_struct_star_9_100 *example dataset for* dependence.structure

---

Description

It was generated by

```
set.seed(222454572)
N = 100
y = coins(N,2)
dep_struct_star_9_100 = cbind(y,y,y)
save(dep_struct_star_9_100,file ="dep_struct_star_9_100.rda")
```

Usage

```
dep_struct_star_9_100
```

Format

matrix 9 variables (columns), 100 independent samples (rows)

Details

To avoid irritation, note that the seed is just a simple integer hash value of the variable name.

`find.cluster` *cluster detection*

Description

Performs the detection of dependence structures algorithm until a cluster is found. This function is the basic building block `dependence.structure`. Advanced users, might use it directly.

Usage

```
find.cluster(x, vec = 1:ncol(x), array.cdm = cdms(x, vec = vec),
  mem = as.numeric(1:max(vec)), cluster.to.vertex = 1:max(mem),
  vertex.to.cdm = 1:max(mem), previous.n.o.cdms = rep(0, max(mem)),
  all.multivariances = numeric(0),
  g = igraph::add.vertices(igraph::graph.empty(, directed = FALSE), max(mem),
  label = sapply(1:max(mem), function(r) paste(colnames(x, do.NULL = FALSE,
  prefix = "")[vec == r], collapse = ",")), shape = "circle"),
  fixed.rejection.level = NA, alpha = 0.05, p.adjust.method = "holm",
  verbose = TRUE, kvec = 2:max(mem))
```

Arguments

`x`	matrix with the samples
`vec`	vector, it indicates which columns are initially treated together as one sample
`array.cdm`	array of centered distance matrices
`mem`	numeric vector, its length is the number of vertices, its content is the number of the corresponding cluster for the current iteration, i.e., vertex `i` belongs to cluster `mem[i]`
`cluster.to.vertex`	vector, contains the cluster to vertex relations, i.e., `cluster.to.vertex[i]` is the index of the vertex which represents cluster `i`
`vertex.to.cdm`	vector, contains the vertex to centered distance matrix relations, i.e., `vertex.to.cdm[i]` is the index centered distance matrix in `array.cdm` which corresponds to vertex `i`
`previous.n.o.cdms`	vector, number of centered distance matrices in the previous iteration (it is used to ensure that previously check tuples are not checked again)
`all.multivariances`	vector, which contains all distance multivariances which have been calculated so far. Only used to finally return all distance multivariances which have been calculated.

g	dependence structure graph fixed.rejection.level = NA, alpha=0.05,method = "holm",explore = FALSE, verbose = TRUE, kvec = 2:max(mem)
fixed.rejection.level	vector, if not NA the fixed.rejection.level[k] is used for the k-tuples, instead of a level derived from the significance level alpha
alpha	numeric, significance level used for the (distribution-free) tests
p.adjust.method	name of the method used to adjust the p-values for multiple testing, see p.adjust for all possible options.
verbose	boolean, if TRUE details during the detection are printed and whenever a cluster is newly detected the (so far) detected dependence structure is plotted.
kvec	vector, k-tuples are only checked for each k in kvec, i.e., for kvec = 2:4 only 2,3 and 4-tuples would be check and then the algorithm stops.

Details

For further details see dependence.structure.

---

independence.test *test for independence*

Description

This computes a test of independence for the columns of a sample matrix (required for the resampling test) or for given centered distance matrices (only possible for the distribution-free test).

Usage

```
independence.test(x, vec = 1:ncol(x), alpha = 0.05,
  type = "distribution_free")
```

Arguments

x	either a data matrix or an array of centered distance matrices
vec	if x is a matrix, then this indicates which columns are treated together as one sample; if x is an array, these are the indexes for which the multivariance is calculated. The default is all columns and all indexes, respectively.
alpha	significance level
type	one of "distribution_free","resample"

Details

The "distribution_free" test might be very conservative. The centered distance matrices can be prepared by cdms. But note that for the resample test, the data matrix has to be given.

Value

Returns TRUE if the hypothesis of independence is NOT rejected, otherwise FALSE.

References

For the theoretic background see the references given on the main help page of this package: multivariance-package.

Examples

```
independence.test(coins(100)) #dependent sample which is 2-independent
independence.test(coins(100),type = "resample") #dependent sample which is 2-independent

independence.test(coins(100)[,2:3]) # independent sample
independence.test(coins(100)[,2:3],type = "resample") # independent sample

independence.test(coins(10),type = "resample") #dependent sample which is 2-independent
independence.test(coins(10)[,2:3],type = "resample") #dependent sample which is 2-independent
```

---

m.multivariance *m distance multivariance*

Description

Computes m distance multivariance.

Usage

```
m.multivariance(x, vec = NA, m = 2, Nscale = TRUE, Escale = TRUE,
  squared = TRUE, ...)
```

Arguments

x	either a data matrix or an array of centered distance matrices
vec	if x is a matrix, then this indicates which columns are treated together as one sample; if x is an array, these are the indexes for which the multivariance is calculated. The default is all columns and all indexes, respectively.
m	=2 or 3 the m-multivariance will be computed.
Nscale	if TRUE the multivariance is scaled up by the sample size (and thus it is exactly as required for the test of independence)
Escale	if TRUE then it is scaled by the number of multivariances which are theoretically summed up (this yields an expectation of 1 in the case of independence)
squared	if FALSE it returns the actual multivariance, otherwise the squared multivariance (less computation)
...	these are passed to cdms (which is only invoked if x is a matrix)

Details

m-distance multivariance is per definition the scaled sum of certain distance multivariances, and it characterize m-dependence.

As a rough guide to interpret the value of total distance multivariance note:

- Large values indicate dependence.
- If the random variables are (m-1)-independent and `Nscale = TRUE`, values close to 1 and smaller indicate m-independence, larger values indicate dependence. In fact, in the case of independence the test statistic is a gaussian quadratic form with expectation 1 and samples of it can be generated by `resample.multivariance`.
- If the random variables are (m-1)-independent and `Nscale = FALSE`, small values (close to 0) indicate m-independence, larger values indicate dependence.

Since random variables are always 1-independent, the case `m=2` characterizes pairwise independence.

Finally note, that due to numerical (in)precision the value of m-multivariance might become negative. In these cases it is set to 0. A warning is issued, if the value is negative and further than the usual (used by `all.equal`) tolerance away from 0.

References

For the theoretic background see the reference [3] given on the main help page of this package: multivariance-package.

Examples

```
x = matrix(rnorm(3*30),ncol = 3)

# the following values are identical
m.multivariance(x,m =2)
1/choose(3,2)*(multivariance(x[,c(1,2)]) +
               multivariance(x[,c(1,3)]) +
               multivariance(x[,c(2,3)]))

# the following values are identical
m.multivariance(x,m=3)
multivariance(x)

# the following values are identical
1/4*(3*(m.multivariance(x,m=2)) + m.multivariance(x,m=3))
total.multivariance(x, Nscale = TRUE)
1/4*(multivariance(x[,c(1,2)], Nscale = TRUE) +
     multivariance(x[,c(1,3)], Nscale = TRUE) +
     multivariance(x[,c(2,3)], Nscale = TRUE) + multivariance(x, Nscale = TRUE))
```

multicorrelation *distance multicorrelation*

Description

computes the distance multicorrelation

Usage

```
multicorrelation(x, vec = NA, squared = FALSE, ...)
```

Arguments

x	either a data matrix or an array of centered distance matrices
vec	if x is a matrix, then this indicates which columns are treated together as one sample; if x is an array, these are the indexes for which the multivariance is calculated. The default is all columns and all indexes, respectively.
squared	if FALSE it returns the actual multivariance, otherwise the squared multivariance (less computation)
...	these are passed to cdms (which is only invoked if x is a matrix)

Details

This is just a wrapper for `multivariance(x,vec,Nscale = FALSE,correlation = TRUE,squared = squared,...)`.

References

For the theoretic background see the references given on the main help page of this package: multivariance-package.

multivariance *distance multivariance*

Description

Computes the distance multivariance, either for given data or a given array of centered distance matrices.

Usage

```
multivariance(x, vec = NA, Nscale = TRUE, correlation = FALSE,
  squared = TRUE, ...)
```

Arguments

x	either a data matrix or an array of centered distance matrices
vec	if x is a matrix, then this indicates which columns are treated together as one sample; if x is an array, these are the indexes for which the multivariance is calculated. The default is all columns and all indexes, respectively.
Nscale	if TRUE the multivariance is scaled up by the sample size (and thus it is exactly as required for the test of independence)
correlation	if TRUE the multivariance is scaled by norms of their centered distance matrices, and Nscale will be ignored.
squared	if FALSE it returns the actual multivariance, otherwise the squared multivariance (less computation)
...	these are passed to cdms (which is only invoked if x is a matrix)

Details

If x is an matrix and vec is not given, then each column is treated as a separate sample. Otherwise vec has to have as many elements as x has columns and values starting from 1 up to the number of 'variables', e.g. if x is an N by 5 matrix and `vec = c(1,2,1,3,1)` then the multivariance of the 1-dimensional variables represented by column 2 and 4 and the 3-dimensional variable represented by the columns 1,3,5 is computed.

As default it computes the normalized Nscaled squared multivariance, for a multivariance without normalization the argument `normalize = FALSE` has to be passed to cdms.

If x is an array, then vec has to be given.

`correlation = TRUE` yields values between 0 and 1. These can be interpreted similarly to classical correlations, see also multicorrelation.

As a rough guide to interpret the value of distance multivariance note:

- If the random variables are not (n-1)-independent, large values indicate dependence, but small values are meaningless. Thus in this case use total.multivariance.
- If the random variables are (n-1)-independent and `Nscale = TRUE`, values close to 1 and smaller indicate independence, larger values indicate dependence. In fact, in the case of independence the test statistic is a Gaussian quadratic form with expectation 1 and samples of it can be generated by resample.multivariance.
- If the random variables are (n-1)-independent and `Nscale = FALSE`, small values (close to 0) indicate independence, larger values indicate dependence.

Finally note, that due to numerical (in)precision the value of multivariance might become negative. In these cases it is set to 0. A warning is issued, if the value is negative and further than the usual (used by all.equal) tolerance away from 0.

References

For the theoretic background see the references given on the main help page of this package: multivariance-package.

Examples

```
multivariance(matrix(rnorm(100*3),ncol = 3)) #independent sample
multivariance(coins(100)) #dependent sample which is 2-independent

x = matrix(rnorm(100*2),ncol = 2)
x = cbind(x,x[,2])
multivariance(x) #dependent sample which is not 2-independent (thus small values are meaningless!)
multivariance(x[,1:2]) #these are independent
multivariance(x[,2:3]) #these are dependent

multivariance(x[,2:3],correlation = TRUE)
```

---

multivariance.pvalue *transform multivariance to p-value*

Description

Computes the p-value for the hypothesis of independence for a given multivariance/total multivariance.

Usage

```
multivariance.pvalue(x)
```

Arguments

x	value of a normalized and Nscaled multivariance

Details

This is based on a distribution-free approach. The p-value is conservative, i.e. it might be much smaller. This is the counterpart to rejection.level. For a less conservative approach see resample.pvalue.

p-values larger than 0.215 might be incorrect, since the distribution-free estimate on which the computation is based only holds up to 0.215.

References

For the theoretic background see the references given on the main help page of this package: multivariance-package.

`multivariances.all` *simultaneous computation of total/ 2-/ 3- /... multivariance*

Description

Computes simultaneously multivariance, total multivariance, 2-multivariance and 3-multivariance.

Usage

```
multivariances.all(x, vec = NA, Nscale = TRUE, squared = TRUE, ...)
```

Arguments

`x`	either a data matrix or an array of centered distance matrices
`vec`	if x is a matrix, then this indicates which columns are treated together as one sample; if x is an array, these are the indexes for which the multivariance is calculated. The default is all columns and all indexes, respectively.
`Nscale`	if `TRUE` the multivariance is scaled up by the sample size (and thus it is exactly as required for the test of independence)
`squared`	if `FALSE` it returns the actual multivariance, otherwise the squared multivariance (less computation)
`...`	these are passed to `cdms` (which is only invoked if x is a matrix)

Details

The computation is faster than the seperate computations.

See Also

`multivariance`, `total.multivariance`, `m.multivariance`

Examples

```
x = coins(100,k = 3)
multivariances.all(x)
# yields the same as:
multivariance(x)
total.multivariance(x)
m.multivariance(x,m=2)
m.multivariance(x,m=3)
```

`rejection.level` *rejection level for the test statistic*

Description

Under independence the probability for the normalized and Nscaled multivariance to be above this level is less than `alpha`. The same holds for the normalized, Nscaled and Escaled total multivariance and m-multivariance.

Usage

```
rejection.level(alpha)
```

Arguments

`alpha`	level of significance

Details

This is based on a distribution-free approach. The value might be very conservative. This is the counterpart to `multivariance.pvalue`. For a less conservative approach see `resample.rejection.level`.

The estimate is only valid for `alpha` smaller than 0.215.

Examples

```
rejection.level(0.05) #the rejection level, for comparison with the following values
total.multivariance(matrix(rnorm(100*3),ncol = 3)) #independent sample
total.multivariance(coins(100)) #dependent sample which is 2-independent

# and the p values are (to compare with alpha)
multivariance.pvalue(total.multivariance(matrix(rnorm(100*3),ncol = 3))) #independent sample
multivariance.pvalue(total.multivariance(coins(100))) #dependent sample which is 2-independent

## Not run:
# visualization of the rejection level
curve(rejection.level(x),xlim = c(0.001,0.215),xlab = "alpha")

## End(Not run)
```

`resample.multivariance` *resampling (total /m-) multivariance*

Description

The distribution of the test statistic under the hypothesis of independence is required for the independence tests. This function generates approximate samples of this distribution either by sampling without replacement (permutations) or by sampling with replacement (bootstrap).

Usage

```
resample.multivariance(x, vec = 1:ncol(x), times = 300, type = "multi",
  resample.type = "permutation", ...)
```

Arguments

`x`	matrix, the rows should be iid samples
`vec`	vector, which indicates which columns of `x` are treated together as one sample
`times`	integer, number of samples to generate
`type`	one of `"multi"`,`"total"`,`"m.multi.2"`,`"m.multi.3"`
`resample.type`	one of `"permutation"`, `"bootstrap"`. The samples are generated without replacement (permutations) or with replacement (bootstrap).
`...`	is passed to `multivariance`, `total.multivariance`, `m.multivariance`, respectively.

Details

The resampling is done by sampling from the original data either without replacement (`"permutation"`) or with replacement (`"bootstrap"`).

For convenience also the actual (total /m-) multivariance is computed and its p-value.

Value

A list with elements

`resampled` the (total/m-)multivariances of the resampled data,

`original` the (total/m-)multivariance of the original data,

`p.value` the p-value of the original data, computed using the resampled data

References

For the theoretic background see the reference [3] given on the main help page of this package: multivariance-package.

Examples

```
re.m = resample.multivariance(matrix(rnorm(30*2),nrow = 30),
                        Nscale = TRUE,type= "multi",times = 300)$resampled
curve(ecdf(re.m)(x), xlim = c(0,4),main = "empirical distribution of the test statistic under H_0")
```

`resample.pvalue` *p-value via resampling*

Description

Use a resampling method to generate samples of the test statistic under the hypothesis of independence. Based on these the p.value of a given value of a test statistic is computed.

Usage

```
resample.pvalue(value, ...)
```

Arguments

`value`	numeric, the value of (total-/m-)multivariance for which the p-value shall be computed
`...`	passed to `resample.multivariance`. Required is the data matrix `x`.

Value

It returns 1 minus the value of the empirical distribution function of the resampling samples evaluated at the given value is returned.

References

For the theoretic background see the reference [3] given on the main help page of this package: multivariance-package.

Examples

```
x = coins(100)
resample.pvalue(multivariance(x),x=x,times = 300)
```

resample.rejection.level *rejection level via resampling*

Description

Uses the resample method to sample from the test statistic under the hypothesis of independence. The alpha quantile of these samples is returned.

Usage

```
resample.rejection.level(alpha = 0.05, ...)
```

Arguments

alpha	numeric, the significance value
...	passed to resample.multivariance. Required is the data matrix x.

References

For the theoretic background see the reference [3] given on the main help page of this package: multivariance-package.

Examples

```
resample.rejection.level(0.05,matrix(rnorm(30*2),nrow = 30))
resample.rejection.level(0.05,matrix(rnorm(30*3),nrow = 30),vec = c(1,1,2))
```

sample.cols *resample the columns of a matrix*

Description

resample the columns of a matrix

Usage

```
sample.cols(x, vec = 1:ncol(x), replace = TRUE)
```

Arguments

x	matrix
vec	vector, indicates which columns belong together
replace	boolean, sampling with or without replacement

Value

Returns a matrix with the same dimensions as `x`. The columns are resampled from the original columns. The resampling is done with replacement (`replace = TRUE`) or without (`replace = FALSE`). Columns which belong together (indicated by vec) are resampled identically, i.e., all values in rows of these are kept together.

Examples

```
sample.cols(matrix(1:15,nrow = 5),vec = c(1,1,2))
```

`tetrahedron` *dependence example: tetrahedron sampling*

Description

This function creates samples of a tetrahedron-dice colored r, g, b and rgb. Each sample indicates if for the thrown dice the colors r, g and b are contained on the bottom side of the dice.

Usage

```
tetrahedron(N = 1000)
```

Arguments

`N`	number of samples

Value

It returns the samples of the events r, g and b as rows of a `N` by 3 matrix (the first column corresponds to r, the second to g,...). TRUE indicates that this color is on the bottom side of the dice. The columns are dependent but 2-independent.

References

For the theoretic background see the reference [3] given on the main help page of this package: multivariance-package.

Examples

```
tetrahedron(10)
```

`total.multivariance` *total distance multivariance*

Description

computes the total distance multivariance

Usage

```
total.multivariance(x, vec = NA, lambda = 1, Nscale = TRUE,
  Escale = TRUE, squared = TRUE, ...)
```

Arguments

`x`	either a data matrix or an array of centered distance matrices
`vec`	if x is a matrix, then this indicates which columns are treated together as one sample; if x is an array, these are the indexes for which the multivariance is calculated. The default is all columns and all indexes, respectively.
`lambda`	a scaling parameter >0. Each k-tuple multivariance gets weight `lambda^(n-k)`.
`Nscale`	if `TRUE` the multivariance is scaled up by the sample size (and thus it is exactly as required for the test of independence)
`Escale`	if `TRUE` then it is scaled by the number of multivariances which are theoretically summed up (in the case of independence this yields for normalized distance matrices an estimator with expectation 1)
`squared`	if `FALSE` it returns the actual multivariance, otherwise the squared multivariance (less computation)
`...`	these are passed to `cdms` (which is only invoked if x is a matrix)

Details

Total distance multivariance is per definition the scaled sum of certain distance multivariances, and it characterize dependence.

As a rough guide to interpret the value of total distance multivariance note:

- Large values indicate dependence.
- For`Nscale = TRUE` values close to 1 and smaller indicate independence, larger values indicate dependence. In fact, in the case of independence the test statistic is a Gaussian quadratic form with expectation 1 and samples of it can be generated by `resample.multivariance`.
- For `Nscale = FALSE` small values (close to 0) indicate independence, larger values indicate dependence.

Finally note, that due to numerical (in)precision the value of total multivariance might become negative. In these cases it is set to 0. A warning is issued, if the value is negative and further than the usual (used by `all.equal`) tolerance away from 0.

References

For the theoretic background see the references given on the main help page of this package: multivariance-package.

Examples

```
x = matrix(rnorm(100*3),ncol = 3)
total.multivariance(x) #for an independent sample
# the value coincides with
(multivariance(x[,c(1,2)],Nscale = TRUE) + multivariance(x[,c(1,3)],Nscale = TRUE)+
 multivariance(x[,c(2,3)],Nscale = TRUE) + multivariance(x,Nscale = TRUE))/4

total.multivariance(coins(100)) #value for a dependent sample which is 2-independent
```

Index