Package ‘MCARtest’

June 26, 2025

Title Optimal Nonparametric Testing of Missing Completely at Random

Version 1.3

Maintainer Thomas B. Berrett <`tom.berrett@warwick.ac.uk`>

Description Provides functions for carrying out nonparametric hypothesis tests of the MCAR hypothesis based on the theory of Frechet classes and compatibility. Also gives functions for computing halfspace representations of the marginal polytope and related geometric objects.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports lpSolve, rcdd, gtools, Epi, Rdpack, Rcpp, pracma, highs, Matrix, Rcsdp, norm, missMethods, copula, MASS

LinkingTo Rcpp

RdMacros Rdpack

NeedsCompilation yes

Author Thomas B. Berrett [aut, cre] (ORCID: <`https://orcid.org/0000-0002-2005-110X`>),
Alberto Bordino [aut],
Danat Duisenbekov [aut],
Sean Jaffe [aut],
Richard J. Samworth [aut] (ORCID: <`https://orcid.org/0000-0003-2426-4679`>)

Repository CRAN

Date/Publication 2025-06-26 09:40:02 UTC

Contents

`Amatrix` *Generate the matrix A, whose columns are the vertices of the marginal polytope.*

Description

Generate the matrix A, whose columns are the vertices of the marginal polytope.

Usage

```
Amatrix(bS, M)
```

Arguments

`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.

Value

The matrix A.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3)
M=c(2,2,2)
Amatrix(bS,M)
```

`AmatrixSparse` *Generate the matrix A, whose columns are the vertices of the marginal polytope, as a sparse matrix.*

Description

Generate the matrix A, whose columns are the vertices of the marginal polytope, as a sparse matrix.

Usage

```
AmatrixSparse(bS, M)
```

Arguments

`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.

Value

The matrix A.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3)
M=c(2,2,2)
AmatrixSparse(bS,M)
```

`aMatrixSparseRevLex` *Generates the row indices used internally to generate the sparse matrix A.*

Description

Generates the row indices used internally to generate the sparse matrix A.

Usage

```
aMatrixSparseRevLex(bS, M)
```

Arguments

`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.

Value

A vector of row indices.

`av` *Compute the columnwise average of a collection of vectors.*

Description

A function that computes $|\mathbb{S}_j|^{-1} \sum_{S \in \mathbb{S}_j} x_{S,j}$ for a collection of vectors $x_{\mathbb{S}}$ over the missingness patterns. This is defined in Step 3 of Algorithms 2 and 3 in Bordino and Berrett (2025).

Usage

```
av(x_S, patterns)
```

Arguments

`x_S`	The collection of vectors over missingness patterns.
`patterns`	The collection of missingness patterns.

Value

The vector of entry $|\mathbb{S}_j|^{-1} \sum_{S \in \mathbb{S}_j} x_{S,j}$.

References

Bordino A, Berrett TB (2025). “Tests of Missing Completely At Random based on sample covariance matrices.” *Ann. Statist., to appear, arXiv:2401.05256.*

Examples

```
library(MASS)

d = 3
n = 200
SigmaS=list() #Random 2x2 correlation matrices (necessarily consistent)
for(j in 1:d){
x=runif(2,min=-1,max=1); y=runif(2,min=-1,max=1); SigmaS[[j]]=cov2cor(x%*%t(x) + y%*%t(y))
}

X = data.frame(matrix(nrow = 3*n, ncol = 3))
X[1:n, c(1,2)] = mvrnorm(n, c(0,0), SigmaS[[1]])
X[(n+1):(2*n), c(2, 3)] = mvrnorm(n, c(0,0), SigmaS[[2]])
X[(2*n+1):(3*n), c(1, 3)] = mvrnorm(n, c(0,0), SigmaS[[3]])
X = as.matrix(X)

tmp = get_SigmaS(X)
av(tmp$mu_S, tmp$patterns)
```

`Cimproved` *Calculate the critical value for our improved test*

Description

Calculate a critical value for an MCAR test based on knowledge of the facet structure of the Minkowski sum calculated by `ConsMinkSumHrep`.

Usage

```
Cimproved(nS, bS, M, DR, Fp, alpha)
```

Arguments

`nS`	A vector of sample sizes, with each entry corresponding to an observation pattern.
`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`DR`	The quantity D_R appearing in Berrett and Samworth (2023).
`Fp`	The quantity F' appearing in Berrett and Samworth (2023).
`alpha`	The desired significance level α of the test.

Value

The critical value C'_α defined in Berrett and Samworth (2023).

References

Berrett TB, Samworth RJ (2023). “Optimal nonparametric testing of Missing Completely At Random, and its connections to compatibility.” *Ann. Statist.*, **51**, 2170–2193.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3)
r=4; s=3
M=c(r,s,2)
Cimproved(rep(1000,3),bS,M,1,(2^r-2)*(2^s-2),0.05)
```

`colVector` *Generates the column indices used internally to generate the sparse matrix A.*

Description

Generates the column indices used internally to generate the sparse matrix A.

Usage

```
colVector(cardS, cardChi)
```

Arguments

`cardS`	The number of missingness patterns.
`cardChi`	The cardinality of the full joint space.

Value

A vector of column indices.

`col_index` *A function indexing the columns of A*

Description

A map from the joint space to an index set.

Usage

```
col_index(M, x)
```

Arguments

`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`x`	An element of the joint space.

Value

A positive integer no greater than the cardinality of the joint space uniquely identifying `x`.

Examples

```
M=c(2,2,2)
col_index(M,c(1,1,1))
col_index(M,c(1,1,2))

M=c(4,3,2)
col_index(M,c(1,1,1))
col_index(M,c(2,1,1))
col_index(M,c(1,2,1))
col_index(M,c(1,1,2))
```

`computeR` *A function computing the incompatibility index for sequences of correlation matrices.*

Description

A function solving a SDP problem to compute the incompatibility index $R()$ for a sequence of correlation matrices, as defined in Bordino and Berrett (2025). Writes the SDP problem in standard primal form, and uses `csdp` to solve this.

Usage

```
computeR(patterns = list(), SigmaS = list())
```

Arguments

`patterns`	A vector with all the patterns in $\mathbb{S}$
`SigmaS`	The sequence of correlation matrices $\Sigma_{\mathbb{S}}$

Value

The value of $R()$, in the interval $[0, 1]$.

The optimal $X_{\mathbb{S}}$ for the primal problem.

The sequence of matrices $X^0_{\mathbb{S}}$ as defined in Bordino and Berrett (2025).

The optimal Σ for the dual problem.

The sequence of correlation matrices $\Sigma_{\mathbb{S}}$ in input.

References

Bordino A, Berrett TB (2025). "Tests of Missing Completely At Random based on sample covariance matrices." *Ann. Statist., to appear, arXiv:2401.05256.*

Examples

```
d = 3

SigmaS=list() #Random 2x2 correlation matrices (necessarily consistent)
for(j in 1:d){
x=runif(2,min=-1,max=1); y=runif(2,min=-1,max=1)
SigmaS[[j]]=cov2cor(x%*%t(x) + y%*%t(y))
}

result = computeR(list(c(1,2),c(2,3), c(1,3)), SigmaS = SigmaS)
result$R
```

`computeR.reg` *A function computing the regularised incompatibility index for collections of correlation matrices.*

Description

A function solving a SDP problem to compute the regularised incompatibility index $R_z()$ for a collection of correlation matrices, as defined in (7) in Bordino and Berrett (2025). Writes the SDP problem in standard primal form, and uses `csdp` to solve this.

Usage

```
computeR.reg(patterns = list(), SigmaS = list(), alpha)
```

Arguments

`patterns`	A vector with all the patterns in $\mathbb{S}$
`SigmaS`	The sequence of correlation matrices $\Sigma_{\mathbb{S}}$
`alpha`	The regularisation parameter, which satisfies alpha = 1/z.

Value

The value of $R_z()$, in the interval $[0, 1]$.

The optimal $X_{\mathbb{S}}$ for the primal problem.

The sequence of matrices $X^0_{\mathbb{S}}$ as defined in Bordino and Berrett (2025).

The optimal Σ for the dual problem.

The sequence of correlation matrices $\Sigma_{\mathbb{S}}$ in input.

References

Bordino A, Berrett TB (2025). “Tests of Missing Completely At Random based on sample covariance matrices.” *Ann. Statist., to appear, arXiv:2401.05256.*

Examples

```
d = 3

SigmaS=list() #Random 2x2 correlation matrices (necessarily consistent)
for(j in 1:d){
x=runif(2,min=-1,max=1); y=runif(2,min=-1,max=1)
SigmaS[[j]]=cov2cor(x%*%t(x) + y%*%t(y))
}

c = 1
for(i in 1:d){
cand = min(eigen(SigmaS[[i]])$values)
if (cand < c){
  c = cand
 }
}

computeR.reg(list(c(1,2),c(2,3), c(1,3)), SigmaS = SigmaS, alpha = 1/c)$R
computeR.reg(list(c(1,2),c(2,3), c(1,3)), SigmaS = SigmaS, alpha = 2)$R
```

`ConsMinkSumHrep` *Calculate the H-representation of the consistent Minkowski sum*

Description

Computes the minimal halfspace representation of the Minkowski sum of the marginal polytope and the consistent ball defined in Berrett and Samworth (2023).

Usage

```
ConsMinkSumHrep(bS, M, round = FALSE)
```

Arguments

`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`round`	A logical value indicating whether or not to round coefficients to 15 significant figures. The function `RoundErrors` can be used separately to substitute other values for 15. Defaults to FALSE.

Value

A halfspace representation object as used by the `rcdd` package. See Geyer and Meeden (2021) for more detail.

References

Berrett TB, Samworth RJ (2023). "Optimal nonparametric testing of Missing Completely At Random, and its connections to compatibility." *Ann. Statist.*, **51**, 2170–2193.

Geyer CJ, Meeden GD (2021). *rcdd: Computational Geometry*. https://CRAN.R-project.org/package=rcdd.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3)
ConsMinkSumHrep(bS,c(2,2,2))
```

`corrCompTest` *Carry out a test of MCAR checking compatibility of correlation matrices.*

Description

This is the implementation of Algorithm 1 in Bordino and Berrett (2025).

Usage

```
corrCompTest(X, B)
```

Arguments

`X`	The dataset with incomplete data.
`B`	The bootstrap sample B for the bootstrap test.

Value

The p-value of the test of MCAR based on correlation matrices, as outlined in Algorithm 1 in Bordino and Berrett (2025).

References

Bordino A, Berrett TB (2025). "Tests of Missing Completely At Random based on sample covariance matrices." *Ann. Statist., to appear, arXiv:2401.05256.*

Examples

```
library(MASS)
alpha = 0.05
B = 20
m = 500

SigmaS=list() #Random 2x2 correlation matrices (necessarily consistent)
for(j in 1:3){
```

```
x=runif(2,min=-1,max=1); y=runif(2,min=-1,max=1)
SigmaS[[j]]=cov2cor(x%*%t(x) + y%*%t(y))
}

X1 = mvrnorm(m, c(0,0), SigmaS[[1]])
X2 = mvrnorm(m, c(0,0), SigmaS[[2]])
X3 = mvrnorm(m, c(0,0), SigmaS[[3]])
columns = c("X1","X2","X3")
X = data.frame(matrix(nrow = 3*m, ncol = 3))
X[1:m, c("X1", "X2")] = X1
X[(m+1):(2*m), c("X2", "X3")] = X2
X[(2*m+1):(3*m), c("X1", "X3")] = X3
X = as.matrix(X)

corrCompTest(X, B)
```

`Csimple` *Calculate the critical value for our simple test*

Description

Calculate a simple critical value for an MCAR test using only knowledge of the set of observation patterns and the joint observation space.

Usage

```
Csimple(nS, bS, M, alpha)
```

Arguments

`nS`	A vector of sample sizes, with each entry corresponding to an observation pattern.
`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`alpha`	The desired significance level α of the test.

Value

The universal critical value defined in Berrett and Samworth (2023).

References

Berrett TB, Samworth RJ (2023). “Optimal nonparametric testing of Missing Completely At Random, and its connections to compatibility.” *Ann. Statist.*, **51**, 2170–2193.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3)
r=4; s=3
M=c(r,s,2)
Csimple(rep(1000,3),bS,M,0.05)
```

`EMiteration` *Perform one step of the EM algorithm for finding the MLE under MCAR in a contingency table.*

Description

Perform one step of the EM algorithm for finding the MLE under MCAR in a contingency table.

Usage

```
EMiteration(pt, p0h, n0, pSh, nS, bS, M)
```

Arguments

`pt`	An input probability mass function on the joint space, to be updated.
`p0h`	An empirical mass function calculated using complete observations.
`n0`	An integer giving the number of complete observations used to calculate `p0h`.
`pSh`	A sequence of empirical mass functions calculated using incomplete observations.
`nS`	A sequence of integers giving the numbers of incomplete observations used to calculate `pSh`.
`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.

Value

The updated probability mass function on the joint space.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3) # Our canonical 3d example
M=c(2,2,2)
n0=200
nS=c(200,200,200)

pS=c(0.125,0.375,0.375,0.125,0.250,0.250,0.250,0.250,0.100,0.400,0.400,0.100)
P12=pS[1:4]; P13=pS[5:8]; P23=pS[9:12]
X12=t(rmultinom(1,size=nS[1],prob=P12)/nS[1])
X13=t(rmultinom(1,size=nS[2],prob=P13)/nS[2])
```

```
X23=t(rmultinom(1,size=nS[3],prob=P23)/nS[3])
pSh=cbind(X12,X13,X23)

p0=array(0.125,dim=c(2,2,2))
p0h=array(rmultinom(1,n0,p0),dim=M)/n0

EMiteration(p0,p0h,n0,pSh,nS,bS,M)
```

`EquivalenceClass` *Simplifies H-representation by exploiting symmetry*

Description

The marginal polytope and related objects have many symmetries. By relabelling the levels of discrete variables we transform facets into other facets. This function reduces a list of halfspace normals to its equivalence classes.

Usage

```
EquivalenceClass(bS, M, Hrep)
```

Arguments

`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`Hrep`	An H-representation generated by `MargPolyHrep`, `ConsMinkSumHrep` or `InconsMinkSumHrep`.

Value

A list of representative halfspace normals.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3) # Our canonical 3d example
Hrep=MargPolyHrep(bS,c(2,2,2))
EquivalenceClass(bS,c(2,2,2),Hrep)
```

`FuchsTest`	*Carry out Fuchs's test of MCAR in a contingency table, given complete and incomplete observations.*

Description

Carry out Fuchs's test of MCAR in a contingency table, given complete and incomplete observations.

Usage

```
FuchsTest(p0h, n0, pSh, nS, bS, M, Niter)
```

Arguments

`p0h`	An empirical mass function calculated using complete observations.
`n0`	An integer giving the number of complete observations used to calculate `p0h`.
`pSh`	A sequence of empirical mass functions calculated using incomplete observations.
`nS`	A sequence of integers giving the numbers of incomplete observations used to calculate `pSh`.
`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`Niter`	An integer giving the number of iterations to be used in the EM algorithm for calculating the null MLE.

Value

The p-value of Fuchs's test, found by comparing the log likelihood ratio statistic to the chi-squared distribution with the appropriate number of degrees of freedom. Described in Fuchs (1982).

References

Fuchs C (1982). "Maximum likelihood estimation and model selection in contingency tables with missing data." *J. Amer. Statist. Assoc.*, **77**(378), 270–278.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3) # Our canonical 3d example
M=c(2,2,2)
n0=200
nS=c(200,200,200)

pS=c(0.125,0.375,0.375,0.125,0.250,0.250,0.250,0.250,0.100,0.400,0.400,0.100)
P12=pS[1:4]; P13=pS[5:8]; P23=pS[9:12]
```

```
X12=t(rmultinom(1,size=nS[1],prob=P12)/nS[1])
X13=t(rmultinom(1,size=nS[2],prob=P13)/nS[2])
X23=t(rmultinom(1,size=nS[3],prob=P23)/nS[3])
pSh=cbind(X12,X13,X23)

p0=array(0.125,dim=c(2,2,2))
p0h=array(rmultinom(1,n0,p0),dim=M)/n0

FuchsTest(p0h,n0,pSh,nS,bS,M,50)
```

`get_SigmaS` *Computes the collection of patterns, means, variances, covariance and correlation matrices for a given dataset with missing values.*

Description

Using the same the notation of Bordino and Berrett (2025), computes the collection of patterns $\mathbb{S}$, means $\mu_{\mathbb{S}}$, variances $\sigma^2_{\mathbb{S}}$, covariance matrices $\Omega_{\mathbb{S}}$ and correlation matrices $\Sigma_{\mathbb{S}}$ for a dataset with missing values.

Usage

```
get_SigmaS(X, min_diff = 0)
```

Arguments

`X`	The dataset with incomplete data.
`min_diff`	A natural number such that patterns with $n_S \leq \lvert S\rvert + min_diff$ are discarded. Default to zero.

Value

`patterns` The collection of patterns $\mathbb{S}$.

`n_pattern` The cardinality of $\mathbb{S}$.

`data_pattern` A vector where the data are grouped according to $\mathbb{S}$.

`mu_S` The collection of means.

`C_S` The collection of covariance matrices.

`sigma_squared_S` The collection of variances.

`SigmaS` The collection of correlation matrices.

`ambient_dimension` The dimension d of the data.

References

Bordino A, Berrett TB (2025). "Tests of Missing Completely At Random based on sample covariance matrices." *Ann. Statist., to appear, arXiv:2401.05256.*

Examples

```
library(copula)
library(missMethods)
n = 100

cp = claytonCopula(param = c(1), dim = 5)
P = mvdc(copula = cp, margins = c("exp", "exp", "exp", "exp", "exp"),
         paramMargins = list(list(1), list(1), list(1), list(1), list(1)))
X = rMvdc(n, P)
X = delete_MCAR(X, 0.1, c(1,4,5))

get_SigmaS(X)
get_SigmaS(X, min_diff = 20)
```

`InconsMinkSumHrep` *Calculate the H-representation of the general (possibly inconsistent) Minkowski sum*

Description

Computes the minimal halfspace representation of the Minkowski sum of the marginal polytope and the inconsistent ball defined in Berrett and Samworth (2023).

Usage

```
InconsMinkSumHrep(bS, M, round = FALSE)
```

Arguments

`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`round`	A logical value indicating whether or not to round coefficients to 15 significant figures. The function `RoundErrors` can be used separately to substitute other values for 15. Defaults to FALSE.

Value

A halfspace representation object as used by the `rcdd` package. See Geyer and Meeden (2021) for more detail.

References

Berrett TB, Samworth RJ (2023). “Optimal nonparametric testing of Missing Completely At Random, and its connections to compatibility.” *Ann. Statist.*, **51**, 2170–2193.

Geyer CJ, Meeden GD (2021). *rcdd: Computational Geometry*. `https://CRAN.R-project.org/package=rcdd`.

Examples

```
bS=matrix(c(1,1, 1,0),byrow=TRUE,ncol=2)
InconsMinkSumHrep(bS,c(2,2))
```

`infoS` *Calculates the total cardinality of the sample spaces.*

Description

Calculates the total cardinality of the sample spaces.

Usage

```
infoS(bS, M)
```

Arguments

`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.

Value

The total cardinality.

`infoS2` *Calculates the individual cardinalities of the sample spaces.*

Description

Calculates the individual cardinalities of the sample spaces.

Usage

```
infoS2(bS, M)
```

Arguments

`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.

Value

A vector of individual cardinalities.

`little_test` *Carry out Little's test of MCAR.*

Description

Carry out Little's test of MCAR.

Usage

```
little_test(X, type = "mean&cov")
```

Arguments

`X`	The dataset with incomplete data, where all the pairs of variables are observed together.
`type`	Determines the test statistic to use, based on the discussion in Section 6 in Bordino and Berrett (2025). The default option is "mean&cov", and uses the test statistic d^2_{aug}. When set equal to "cov", implements a test of MCAR based on d^2_{cov}, while, when set equal to "mean", implements the classical Little's test as defined in Little (1988).

Value

The p-value of Little's test, found by comparing the log likelihood ratio statistic to the chi-squared distribution with the appropriate number of degrees of freedom. Described in Little (1988).

References

Bordino A, Berrett TB (2025). "Tests of Missing Completely At Random based on sample covariance matrices." *Ann. Statist., to appear, arXiv:2401.05256*.

Little RJ (1988). "A test of Missing Completely at Random for multivariate data with missing values." *J. Amer. Statist. Assoc.*, **83**, 1198–1202.

Examples

```
library(MASS)
n = 200

SigmaS=list() #Random 2x2 correlation matrices (necessarily consistent)
for(j in 1:3){
x=runif(2,min=-1,max=1); y=runif(2,min=-1,max=1)
SigmaS[[j]]=cov2cor(x%*%t(x) + y%*%t(y))
}

X1 = mvrnorm(n, c(0,0), SigmaS[[1]])
X2 = mvrnorm(n, c(0,0), SigmaS[[2]])
X3 = mvrnorm(n, c(0,0), SigmaS[[3]])
columns = c("X1","X2","X3")
```

```
X = data.frame(matrix(nrow = 3*n, ncol = 3))
X[1:n, c("X1", "X2")] = X1
X[(n+1):(2*n), c("X2", "X3")] = X2
X[(2*n+1):(3*n), c("X1", "X3")] = X3
X = as.matrix(X)

little_test(X)
little_test(X, type = "mean&cov")
little_test(X, type = "mean")
```

`loglik0` *Compute the log likelihood of a probability mass function, under MCAR, given complete and incomplete data*

Description

Compute the log likelihood of a probability mass function, under MCAR, given complete and incomplete data

Usage

```
loglik0(p, p0h, n0, pSh, nS, bS, M)
```

Arguments

`p`	A probability mass function whose log likelihood is to be calculated.
`p0h`	An empirical mass function calculated using complete observations.
`n0`	An integer giving the number of complete observations used to calculate `p0h`.
`pSh`	A sequence of empirical mass functions calculated using incomplete observations.
`nS`	A sequence of integers giving the numbers of incomplete observations used to calculate `pSh`.
`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.

Value

The value of the log likelihood.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3) # Our canonical 3d example
M=c(2,2,2)
n0=200
nS=c(200,200,200)

pS=c(0.125,0.375,0.375,0.125,0.250,0.250,0.250,0.250,0.100,0.400,0.400,0.100)
P12=pS[1:4]; P13=pS[5:8]; P23=pS[9:12]
X12=t(rmultinom(1,size=nS[1],prob=P12)/nS[1])
X13=t(rmultinom(1,size=nS[2],prob=P13)/nS[2])
X23=t(rmultinom(1,size=nS[3],prob=P23)/nS[3])
pSh=cbind(X12,X13,X23)

p0=array(0.125,dim=c(2,2,2))
p0h=array(rmultinom(1,n0,p0),dim=M)/n0

loglik0(p0,p0h,n0,pSh,nS,bS,M)
```

`loglik1` *Compute the log likelihood of a probability mass function, without assuming MCAR, given complete and incomplete data*

Description

Compute the log likelihood of a probability mass function, without assuming MCAR, given complete and incomplete data

Usage

```
loglik1(p0, pS, p0h, n0, pSh, nS, bS, M)
```

Arguments

`p0`	A probability mass function on the joint space.
`pS`	A sequence of probability mass functions on the marginal spaces.
`p0h`	An empirical mass function calculated using complete observations.
`n0`	An integer giving the number of complete observations used to calculate `p0h`.
`pSh`	A sequence of empirical mass functions calculated using incomplete observations.
`nS`	A sequence of integers giving the numbers of incomplete observations used to calculate `pSh`.
`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.

Value

The value of the log likelihood.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3) # Our canonical 3d example
M=c(2,2,2)
n0=200
nS=c(200,200,200)

pS=c(0.125,0.375,0.375,0.125,0.250,0.250,0.250,0.250,0.100,0.400,0.400,0.100)
P12=pS[1:4]; P13=pS[5:8]; P23=pS[9:12]
X12=t(rmultinom(1,size=nS[1],prob=P12)/nS[1])
X13=t(rmultinom(1,size=nS[2],prob=P13)/nS[2])
X23=t(rmultinom(1,size=nS[3],prob=P23)/nS[3])
pSh=cbind(X12,X13,X23)

p0=array(0.125,dim=c(2,2,2))
p0h=array(rmultinom(1,n0,p0),dim=M)/n0

loglik1(p0,pS,p0h,n0,pSh,nS,bS,M)
```

`M` *Computes an inconsistency index for a collection of means.*

Description

A function that computes the inconsistency index $M(\mu_{\mathbb{S}})$ for a collection of means, as defined in Algorithm 2 in Section 6 of Bordino and Berrett (2025).

Usage

```
M(mu_S, patterns)
```

Arguments

`mu_S`	A list of means $\mu_{\mathbb{S}}$.
`patterns`	A list of missingness patterns.

Value

The value of $M()$.

References

Bordino A, Berrett TB (2025). “Tests of Missing Completely At Random based on sample covariance matrices.” *Ann. Statist., to appear, arXiv:2401.05256.*

Examples

```
library(MASS)

d = 3
n = 200
SigmaS=list() #Random 2x2 correlation matrices (necessarily consistent)
for(j in 1:d){
x=runif(2,min=-1,max=1); y=runif(2,min=-1,max=1); SigmaS[[j]]=cov2cor(x%*%t(x) + y%*%t(y))
}

X = data.frame(matrix(nrow = 3*n, ncol = 3))
X[1:n, c(1,2)] = mvrnorm(n, c(0,0), SigmaS[[1]])
X[(n+1):(2*n), c(2, 3)] = mvrnorm(n, c(0,0), SigmaS[[2]])
X[(2*n+1):(3*n), c(1, 3)] = mvrnorm(n, c(0,0), SigmaS[[3]])
X = as.matrix(X)

tmp = get_SigmaS(X)
M(tmp$mu_S, tmp$patterns)
```

`MargPolyHrep` *Calculate the H-representation of the marginal polytope*

Description

Computes the minimal halfspace representation of the marginal polytope defined, for example, in Berrett and Samworth (2023).

Usage

```
MargPolyHrep(bS, M, round = FALSE)
```

Arguments

`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`round`	A logical value indicating whether or not to round coefficients to 15 significant figures. The function `RoundErrors` can be used separately to substitute other values for 15. Defaults to FALSE.

Value

A halfspace representation object as used by the `rcdd` package. See Geyer and Meeden (2021) for more detail.

References

Berrett TB, Samworth RJ (2023). "Optimal nonparametric testing of Missing Completely At Random, and its connections to compatibility." *Ann. Statist.*, **51**, 2170–2193.

Geyer CJ, Meeden GD (2021). *rcdd: Computational Geometry*. https://CRAN.R-project.org/package=rcdd.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3)
MargPolyHrep(bS,c(2,2,2))
```

`margProj` *Internal function multiplying a mass function by the sparse matrix A.*

Description

Internal function multiplying a mass function by the sparse matrix A.

Usage

```
margProj(p, bS, M)
```

Arguments

`p`	A subprobability mass function on the full joint space.
`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.

Value

A collection of marginal mass functions.

`meanConsTest` *Carry out a test of MCAR checking consistency of mean vectors.*

Description

This is the implementation of Algorithm 2 in Bordino and Berrett (2025).

Usage

```
meanConsTest(X, B)
```

Arguments

`X`	The dataset with incomplete data.
`B`	The bootstrap sample B for the bootstrap test.

Value

The p-value of the test of MCAR based on mean vectors, as outlined in Algorithm 2 in Bordino and Berrett (2025).

References

Bordino A, Berrett TB (2025). “Tests of Missing Completely At Random based on sample covariance matrices.” *Ann. Statist., to appear, arXiv:2401.05256*.

Examples

```
library(MASS)
alpha = 0.05
B = 20
m = 500

SigmaS=list() #Random 2x2 correlation matrices (necessarily consistent)
for(j in 1:3){
x=runif(2,min=-1,max=1); y=runif(2,min=-1,max=1)
SigmaS[[j]]=cov2cor(x%*%t(x) + y%*%t(y))
}

X1 = mvrnorm(m, c(1,0), SigmaS[[1]])
X2 = mvrnorm(m, c(0,0), SigmaS[[2]])
X3 = mvrnorm(m, c(3,0), SigmaS[[3]])
columns = c("X1","X2","X3")
X = data.frame(matrix(nrow = 3*m, ncol = 3))
X[1:m, c("X1", "X2")] = X1
X[(m+1):(2*m), c("X2", "X3")] = X2
X[(2*m+1):(3*m), c("X1", "X3")] = X3
X = as.matrix(X)

meanConsTest(X, B)
```

`MLE` *Compute the MLE under MCAR in a contingency table using the EM algorithm, given complete and incomplete observations.*

Description

Compute the MLE under MCAR in a contingency table using the EM algorithm, given complete and incomplete observations.

Usage

```
MLE(p0h, n0, pSh, nS, bS, M, Niter, loglik = FALSE)
```

Arguments

`p0h`	An empirical mass function calculated using complete observations.
`n0`	An integer giving the number of complete observations used to calculate `p0h`.
`pSh`	A sequence of empirical mass functions calculated using incomplete observations.
`nS`	A sequence of integers giving the numbers of incomplete observations used to calculate `pSh`.
`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`Niter`	An integer giving the number of iterations to be used in the EM algorithm.
`loglik`	A logical value indicating whether or not the log likelihoods at each step of the EM algorithm should be an output. Defaults to FALSE.

Value

The output of the EM algorithm, approximating the MLE for the probability mass function on the joint space.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3) # Our canonical 3d example
M=c(2,2,2)
n0=200
nS=c(200,200,200)

pS=c(0.125,0.375,0.375,0.125,0.250,0.250,0.250,0.250,0.100,0.400,0.400,0.100)
P12=pS[1:4]; P13=pS[5:8]; P23=pS[9:12]
X12=t(rmultinom(1,size=nS[1],prob=P12)/nS[1])
X13=t(rmultinom(1,size=nS[2],prob=P13)/nS[2])
X23=t(rmultinom(1,size=nS[3],prob=P23)/nS[3])
pSh=cbind(X12,X13,X23)

p0=array(0.125,dim=c(2,2,2))
p0h=array(rmultinom(1,n0,p0),dim=M)/n0

MLE(p0h,n0,pSh,nS,bS,M,50)

trace=MLE(p0h,n0,pSh,nS,bS,M,50,loglik=TRUE)[[2]]
plot(1:50,trace,type="l")
```

`ProjectionTest` *Carry out a test of MCAR in a contingency table, given incomplete observations.*

Description

Carry out a test of MCAR in a contingency table, given incomplete observations.

Usage

```
ProjectionTest(pSh, nS, bS, M, B)
```

Arguments

`pSh`	A sequence of empirical mass functions calculated using incomplete observations.
`nS`	A sequence of integers giving the numbers of incomplete observations used to calculate `pSh`.
`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`B`	An integer giving the number of bootstrap samples to be used to calibrate the test.

Value

The p-value the Monte Carlo test described in Berrett and Samworth (2023).

The value of the test statistic $R()$.

References

Berrett TB, Samworth RJ (2023). "Optimal nonparametric testing of Missing Completely At Random, and its connections to compatibility." *Ann. Statist.*, **51**, 2170–2193.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3) # Our canonical 3d example
M=c(2,2,2)
nS=c(200,200,200)

pS=c(0.125,0.375,0.375,0.125,0.250,0.250,0.250,0.250,0.100,0.400,0.400,0.100)
P12=pS[1:4]; P13=pS[5:8]; P23=pS[9:12]
X12=t(rmultinom(1,size=nS[1],prob=P12)/nS[1])
X13=t(rmultinom(1,size=nS[2],prob=P13)/nS[2])
X23=t(rmultinom(1,size=nS[3],prob=P23)/nS[3])
pSh=cbind(X12,X13,X23)

ProjectionTest(pSh,nS,bS,M,99)
```

`Rindex` *A function computing the incompatibility index*

Description

A function solving a linear program to compute the incompatibility index $R()$ defined in Berrett and Samworth (2023), in the case of having discrete random variables. Uses `Amatrix` to define to constraint matrix and `lpSolve` to implement the linear optimisation.

Usage

```
Rindex(pS, bS, M)
```

Arguments

`pS`	A sequence of probability mass functions on the marginal spaces.
`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.

Value

The value of $R()$, in the interval $[0, 1]$.

References

Berrett TB, Samworth RJ (2023). "Optimal nonparametric testing of Missing Completely At Random, and its connections to compatibility." *Ann. Statist.*, **51**, 2170–2193.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3)
M=c(2,2,2)

pS=rep(0.25,12)
Rindex(pS,bS,M)

pS=c(0.125,0.375,0.375,0.125,0.250,0.250,0.250,0.250,0.100,0.400,0.400,0.100)
Rindex(pS,bS,M)
```

`RindexDual`	*A function computing the incompatibility index and associated closest joint mass function using the dual formulation*

Description

A function solving a linear program to compute the incompatibility index $R()$ defined in Berrett and Samworth (2023), in the case of having discrete random variables. Uses `Amatrix` to define to constraint matrix and `lpSolve` to implement the linear optimisation.

Usage

```
RindexDual(pS, bS, M, lp_solver = "default", simplex_strategy = 4)
```

Arguments

`pS`	A sequence of probability mass functions on the marginal spaces.
`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`lp_solver`	An argument passed to HiGHS specifying which solver to use.
`simplex_strategy`	An argument passed to HiGHS specifying which solver to use.

Value

The value of $R()$, in the interval $[0, 1]$.

The optimal solution to the linear program

References

Berrett TB, Samworth RJ (2023). "Optimal nonparametric testing of Missing Completely At Random, and its connections to compatibility." *Ann. Statist.*, **51**, 2170–2193.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3)
M=c(2,2,2)
A=Amatrix(bS,M)

pS=rep(0.25,12)
linprog=RindexDual(pS,bS,M)
rbind(pS,as.vector(A%*%linprog[[2]])/(1-linprog[[1]]))

pS=c(0.125,0.375,0.375,0.125,0.250,0.250,0.250,0.250,0.100,0.400,0.400,0.100)
linprog=RindexDual(pS,bS,M)
rbind(pS,as.vector(A%*%linprog[[2]])/(1-linprog[[1]]))
```

`RoundErrors` *Round errors in halfspace representations*

Description

Round errors in halfspace representations

Usage

```
RoundErrors(X, digits = 15)
```

Arguments

`X`	A halfspace representation to be rounded.
`digits`	An integer giving the number of significant figures to be kept.

Value

A rounded halfspace representation.

Examples

```
bS=matrix(c(1,1,1,0, 1,0,0,1, 0,1,0,1, 0,0,1,1),byrow=TRUE,ncol=4)
RoundErrors(”9007199254740992/6004799503160661”) #Occurs in ConsMinkSumHrep(bS,c(2,2,2,2))
```

`row_index` *A function indexing the rows of A*

Description

A map from the observation space to an index set.

Usage

```
row_index(bS, M, S, xS)
```

Arguments

`bS`	A binary matrix specifying the set of observation patterns. Each row encodes a single pattern.
`M`	A vector of positive integers giving the alphabet sizes of the discrete variables.
`S`	An integer indicating which observation pattern is of interest.
`xS`	An element of the observation space of the specified observation pattern.

Value

A positive integer no larger than the cardinality of the joint space uniquely identifying `x`.

Examples

```
bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3)
M=c(2,2,2)
row_index(bS,M,1,c(1,1))
row_index(bS,M,2,c(1,1))
row_index(bS,M,3,c(1,1))

bS=matrix(c(1,1,0, 1,0,1, 0,1,1),byrow=TRUE,ncol=3)
M=c(4,3,2)
row_index(bS,M,1,c(1,1))
row_index(bS,M,1,c(2,1))
row_index(bS,M,1,c(3,1))
row_index(bS,M,1,c(4,1))
row_index(bS,M,1,c(1,2))
row_index(bS,M,1,c(2,2))
```

`V` *Computes an inconsistency index for sequences of variances.*

Description

A function that computes the inconsistency index $V(\sigma^2_{\mathbb{S}})$ for a collection of variances as defined in Algorithm 3 in Section 6 of Bordino and Berrett (2025). Assumes that $|\mathbb{S}_j|^{-1} \sum_{S \in \mathbb{S}_j} \sigma^2_{S,j} = 1$ for all $j \in [d]$.

Usage

```
V(sigma_squared_S, patterns)
```

Arguments

`sigma_squared_S`	A list of variances $\sigma^2_{\mathbb{S}}$.
`patterns`	A list of missingness patterns $\mathbb{S}$.

Value

The value of $V()$, in the interval $[0, 1]$.

References

Bordino A, Berrett TB (2025). “Tests of Missing Completely At Random based on sample covariance matrices.” *Ann. Statist., to appear, arXiv:2401.05256.*

Examples

```
library(MASS)

d = 3
n = 200
SigmaS=list() #Random 2x2 correlation matrices (necessarily consistent)
for(j in 1:d){
x=runif(2,min=-1,max=1); y=runif(2,min=-1,max=1); SigmaS[[j]]=cov2cor(x%*%t(x) + y%*%t(y))
}

X = data.frame(matrix(nrow = 3*n, ncol = 3))
X[1:n, c(1,2)] = mvrnorm(n, c(0,0), SigmaS[[1]])
X[(n+1):(2*n), c(2, 3)] = mvrnorm(n, c(0,0), SigmaS[[2]])
X[(2*n+1):(3*n), c(1, 3)] = mvrnorm(n, c(0,0), SigmaS[[3]])
X = as.matrix(X)

tmp = get_SigmaS(X)
av_sigma = av(tmp$sigma_squared_S, tmp$patterns)
X_new = X
for (j in 1:3){
  X_new[,j] = X[,j]/sqrt(av_sigma[j])
  }

V(get_SigmaS(X_new)$sigma_squared_S, tmp$patterns)
```

`varConsTest` *Carry out a test of MCAR checking consistency of variance vectors.*

Description

This is the implementation of Algorithm 3 in Bordino and Berrett (2025).

Usage

```
varConsTest(X, B)
```

Arguments

`X`	The dataset with incomplete data.
`B`	The bootstrap sample B for the bootstrap test.

Value

The p-value of the test of MCAR based on variance vectors, as outlined in Algorithm 3 in Bordino and Berrett (2025).

References

Bordino A, Berrett TB (2025). “Tests of Missing Completely At Random based on sample covariance matrices.” *Ann. Statist., to appear, arXiv:2401.05256.*

Examples

```
library(MASS)
alpha = 0.05
B = 20
m = 500

SigmaS=list() #Random 2x2 correlation matrices (necessarily consistent)
for(j in 1:3){
x=runif(2,min=-1,max=1); y=runif(2,min=-1,max=1)
SigmaS[[j]]=cov2cor(x%*%t(x) + y%*%t(y))
}

X1 = mvrnorm(m, c(0,0), SigmaS[[1]])
X2 = mvrnorm(m, c(0,0), SigmaS[[2]])
X3 = mvrnorm(m, c(0,0), SigmaS[[3]])
columns = c("X1","X2","X3")
X = data.frame(matrix(nrow = 3*m, ncol = 3))
X[1:m, c("X1", "X2")] = X1
X[(m+1):(2*m), c("X2", "X3")] = X2
X[(2*m+1):(3*m), c("X1", "X3")] = X3
X = as.matrix(X)

varConsTest(X, B)
```

Index