Package ‘CovTools’

November 26, 2019

Type Package
Title Statistical Tools for Covariance Analysis
Version 0.5.3
Description Covariance is of universal prevalence across various disciplines within statistics. We provide a rich collection of geometric and inferential tools for convenient analysis of covariance structures, topics including distance measures, mean covariance estimator, covariance hypothesis test for one-sample and two-sample cases, and covariance estimation. For an introduction to covariance in multivariate statistical analysis, see Schervish (1987) <doi:10.1214/ss/1177013111>.

License GPL (>= 3)
Encoding UTF-8
LazyData true
Depends R (>= 2.14.0)
Imports Rcpp, geigen, shapes, expm, mvtnorm, stats, Matrix, doParallel, foreach, parallel, pracma, Rdpack, utils, SHT
LinkingTo Rcpp, RcppArmadillo
RoxygenNote 7.0.0
RdMacros Rdpack
NeedsCompilation yes
Author Kyoungjae Lee [aut], Lizhen Lin [ctb], Kisung You [aut, cre] (<https://orcid.org/0000-0002-8584-459X>)
Maintainer Kisung You <kyoustat@gmail.com>
Repository CRAN
Date/Publication 2019-11-26 06:40:17 UTC

R topics documented:

BCovTest1.mxPBF ............................................. 2
BDiagTest1.mxPBF ........................................... 4
It performs Bayesian version of 1-sample test for Covariance where the null hypothesis is

\[ H_0 : \Sigma_n = \Sigma_0 \]

where \( \Sigma_n \) is the covariance of data model and \( \Sigma_0 \) is a hypothesized covariance. Denote \( X_i \) be the \( i \)-th column of data matrix. Under the maximum pairwise Bayes Factor framework, we have following hypothesis,

\[ H_0 : a_{ij} = 0 \text{ and } \tau_{ij} = 1 \text{ versus } H_1 : \text{not } H_0. \]

The model is

\[ X_i | X_j \sim N_n(a_{ij} X_j, \tau_{ij}^2 I_n) \]

and the prior is set, under \( H_1 \), as

\[ a_{ij} | \tau_{ij}^2 \sim N(0, \tau_{ij}^2 / (\gamma * \|X_j\|^2)) \]

\[ \tau_{ij}^2 \sim IG(a_0, b_0). \]
Usage

BCovTest1.mxPBF(data, Sigma0 = diag(ncol(data)), a0 = 2, b0 = 2, gamma = 1)

Arguments

data an \( (n \times p) \) data matrix where each row is an observation.
Sigma0 a \( (p \times p) \) given covariance matrix.
a0 shape parameter for inverse-gamma prior.
b0 scale parameter for inverse-gamma prior.
gamma non-negative number. See the equation above.

Value

a named list containing:

log.BF.mat a \( (p \times p) \) matrix of pairwise log Bayes factors.

References


Examples

```r
## Not run:
## generate data from multivariate normal with trivial covariance.
pdim = 10
data = matrix(rnorm(100*pdim), nrow=100)

## run mxPBF-based test
out1 = B CovTest1.mxpBF(data)
out2 = B CovTest1.mxpBF(data, a0=5.0, b0=5.0) # change some params

## visualize two Bayes Factor matrices
opar <- par(mfrow=c(1,2), pty="s")
image(exp(out1$log.BF.mat)[,pdim:1], main="default")
image(exp(out2$log.BF.mat)[,pdim:1], main="a0=b0=5.0")
par(opar)

## End(Not run)
```
One-Sample Diagonality Test by Maximum Pairwise Bayes Factor

Description

One-sample diagonality test can be stated with the null hypothesis

\[ H_0 : \sigma_{ij} = 0 \text{ for any } i \neq j \]

and alternative hypothesis \( H_1 : \text{not } H_0 \) with \( \Sigma_n = (\sigma_{ij}) \). Let \( X_i \) be the \( i \)-th column of data matrix. Under the maximum pairwise bayes factor framework, we have following hypothesis,

\[ H_0 : a_{ij} = 0 \text{ versus } H_1 : \text{not } H_0. \]

The model is

\[ X_i|X_j \sim N_n(a_{ij}X_j, \tau_{ij}^2 I_n). \]

Under \( H_0 \), the prior is set as

\[ \tau_{ij}^2 \sim IG(a_0, b_0) \]

and under \( H_1 \), priors are

\[ a_{ij}|\tau_{ij}^2 \sim N(0, \tau_{ij}^2/(\gamma * ||X_j||^2)) \]
\[ \tau_{ij}^2 \sim IG(a_0, b_0). \]

Usage

BDiagTest1.mxPBF(data, a0 = 2, b0 = 2, gamma = 1)

Arguments

data an \((n \times p)\) data matrix where each row is an observation.
a0 shape parameter for inverse-gamma prior.
b0 scale parameter for inverse-gamma prior.
gamma non-negative number. See the equation above.

Value

da named list containing:

log.BF.mat \((p \times p)\) matrix of pairwise log Bayes factors.

References

## Not run:
## generate data from multivariate normal with trivial covariance.
pdim = 10
data = matrix(rnorm(100*pdim), nrow=100)

## run test
## run mxPBF-based test
out1 = BDiagTest1.mxPBF(data)
out2 = BDiagTest1.mxPBF(data, a0=5.0, b0=5.0) # change some params

## visualize two Bayes Factor matrices
opar <- par(mfrow=c(1,2), pty="s")
image(exp(out1$log.BF.mat)[,pdim:1], main="default")
image(exp(out2$log.BF.mat)[,pdim:1], main="a0=b0=5.0")
par(opar)

## End(Not run)

---

### CovDist

#### Compute Pairwise Distance for Symmetric Positive-Definite Matrices

**Description**

For a given 3-dimensional array where symmetric positive definite (SPD) matrices are stacked slice by slice, it computes pairwise distance using various popular measures. Some of measures are **metric** as they suffice 3 conditions in mathematical context; nonnegative definiteness, symmetry, and triangle inequalities. Other non-metric measures represent **dissimilarities** between two SPD objects.

**Usage**

```r
```

**Arguments**

- `A` a \((p \times p \times N)\) 3d array of \(N\) SPD matrices.
- `method` the type of distance measures to be used;
  - "A" (AIRM) Affine Invariant Riemannian Metric
  - "B" (Bhattacharyya) Bhattacharyya distance based on normal model
  - "C" (Cholesky) Cholesky difference in Frobenius norm
"E" (Euclidean) naïve Frobenius norm as distance
"H" (Hellinger) Hellinger distance based on normal model
"J" (JBLD) Jensen-Bregman Log Determinant Distance
"K" (KLDM) symmetrized Kullback-Leibler Distance Measure
"L" (LERM) Log Euclidean Riemannian Metric
"PS" (Procrustes.SS) Procrustes Size and Shape measure
"PF" (Procrustes.Full) Procrustes analysis with scale
"PE" (PowerEuclidean) weighted eigenvalues by some exponent
"RE" (RootEuclidean) matrix square root

power a non-zero number for PowerEuclidean distance.
as.dist a logical; TRUE to return a dist object, FALSE otherwise.

Value
an \((N \times N)\) symmetric matrix of pairwise distances or a dist object via as.dist option.

References


Examples
## generate 50 SPD matrices of size (5-by-5)
samples = samplecovs(50,5)

## get pairwise distance for several methods
distA = CovDist(samples, method="A")
distB = CovDist(samples, method="B")
distC = CovDist(samples, method="C")

## dimension reduction using MDS
ssA = stats::cmdscale(distA, k=2)
ssB = stats::cmdscale(distB, k=2)
ssC = stats::cmdscale(distC, k=2)

## visualize
opar <- par(mfrow=c(1,3), pty="s")
plot(ssA ,main="project with AIRM", pch=19)
plot(ssB ,main="project with Bhattacharyya", pch=19)
plot(ssC ,main="project with Cholesky", pch=19)
par(opar)
CovEst.2003LW

Covariance Estimation with Linear Shrinkage

Description

Ledoit and Wolf (2003, 2004) proposed a linear shrinkage strategy to estimate covariance matrix with an application to portfolio optimization. An optimal covariance is written as a convex combination as follows,

$$\hat{\Sigma} = \delta \hat{F} + (1 - \delta) \hat{S}$$

where $\delta \in (0, 1)$ a control parameter/weight, $\hat{S}$ an empirical covariance matrix, and $\hat{F}$ a target matrix. Although authors used $F$ a highly structured estimator, we also enabled an arbitrary target matrix to be used as long as it’s symmetric and positive definite of corresponding size.

Usage

CovEst.2003LW(X, target = NULL)

Arguments

- **X**: an $(n \times p)$ matrix where each row is an observation.
- **target**: target matrix $F$. If target=NULL, constant correlation model estimator is used. If target is specified as a qualified matrix, it is used instead.

Value

A named list containing:

- **S**: a $(p \times p)$ covariance matrix estimate.
- **delta**: an estimate for convex combination weight according to the relevant theory.

References


Examples

## CRAN-purpose small computation
# set a seed for reproducibility
set.seed(11)

# small data with identity covariance
pdim <- 5
dat.small <- matrix(rnorm(20*pdim), ncol=pdim)

# run the code with highly structured estimator
out.small <- CovEst.2003LW(dat.small)

# visualize
opar <- par(mfrow=c(1,3), pty="s")
image(diag(5)[,pdim:1], main="true cov")
image(cov(dat.small)[,pdim:1], main="sample cov")
image(out.small$S[,pdim:1], main="estimated cov")
par(opar)

## Not run:
## want to see how delta is determined according to
# the number of observations we have.
nsamples = seq(from=5, to=200, by=5)
nnsample = length(nsamples)

# we will record two values; delta and norm difference
vec.delta = rep(0, nnsample)
vec.normd = rep(0, nnsample)
for (i in 1:nnsample)
    dat.norun <- matrix(rnorm(nsamples[i]*pdim), ncol=pdim) # sample in R^5
    out.norun <- CovEst.2003LW(dat.norun) # run with default
    vec.delta[i] = out.norun$delta
    vec.normd[i] = norm(out.norun$S - diag(pdim),"f") # Frobenius norm

# let's visualize the results
par(mfrow=c(1,2))
plot(nsamples, vec.delta, lwd=2, type="b", col="red", main="estimated deltas")
plot(nsamples, vec.normd, lwd=2, type="b", col="blue", main="Frobenius error")

## End(Not run)
**Description**

Authors propose to estimate covariance matrix by iteratively approximating the shrinkage with

\[ \hat{\Sigma} = \rho \hat{F} + (1 - \rho) \hat{S} \]

where \( \rho \in (0, 1) \) a control parameter/weight, \( \hat{S} \) an empirical covariance matrix, and \( \hat{F} \) a target matrix. It is proposed to use a structured estimate \( \hat{F} = \text{Tr}(\hat{S}/p) \cdot I_{p \times p} \) where \( I_{p \times p} \) is an identity matrix of dimension \( p \).

**Usage**

\[ \text{CovEst.2010OAS}(X) \]

**Arguments**

\( X \) an \((n \times p)\) matrix where each row is an observation.

**Value**

a named list containing:

- \( S \) a \((p \times p)\) covariance matrix estimate.
- \( \text{rho} \) an estimate for convex combination weight.

**References**


**Examples**

```r
## CRAN-purpose small computation
# set a seed for reproducibility
set.seed(11)

# small data with identity covariance
pdim <- 5
dat.small <- matrix(rnorm(10*pdim), ncol=pdim)

# run the code
out.small <- CovEst.2010OAS(dat.small)

# visualize
opar <- par(mfrow=c(1,3), pty="s")
image(diag(pdim)[,pdim:1], main="true cov")
image(cov(dat.small)[,pdim:1], main="sample cov")
image(out.small$S[,pdim:1], main="estimated cov")
par(opar)
```

## Not run:
## want to see how delta is determined according to
# the number of observations we have.
nsamples = seq(from=5, to=200, by=5)
nnsample = length(nsamples)

# we will record two values; rho and norm difference
vec.rho = rep(0, nnsample)
vec.normd = rep(0, nnsample)
for (i in 1:nnsample){
  dat.norun <- matrix(rnorm(nsamples[i]*pdim), ncol=pdim) # sample in R^5
  out.norun <- CovEst.2010OAS(dat.norun) # run with default
  vec.rho[i] = out.norun$rho
  vec.normd[i] = norm(out.norun$S - diag(pdim),"f") # Frobenius norm
}

# let's visualize the results
opar <- par(mfrow=c(1,2))
plot(nsamples, vec.rho, lwd=2, type="b", col="red", main="estimated rhos")
plot(nsamples, vec.normd, lwd=2, type="b", col="blue",main="Frobenius error")
par(opar)

## End(Not run)

---

### CovEst.2010RBLW

**Rao-Blackwell Ledoit-Wolf Estimator**

#### Description

Authors propose to estimate covariance matrix by minimizing mean squared error with the following formula,

\[
\hat{\Sigma} = \rho \hat{F} + (1 - \rho) \hat{S}
\]

where \( \rho \in (0, 1) \) a control parameter/weight, \( \hat{S} \) an empirical covariance matrix, and \( \hat{F} \) a target matrix. It is proposed to use a structured estimate \( \hat{F} = \text{Tr}(\hat{S}/p) \cdot I_{p \times p} \) where \( I_{p \times p} \) is an identity matrix of dimension \( p \).

#### Usage

CovEst.2010RBLW(X)

#### Arguments

- **X**  
an \((n \times p)\) matrix where each row is an observation.

#### Value

- **S**  
a \((p \times p)\) covariance matrix estimate.
- **rho**  
an estimate for convex combination weight.
References


Examples

```r
## CRAN-purpose small computation
# set a seed for reproducibility
set.seed(11)

# small data with identity covariance
cov <- 10
dat.small <- matrix(rnorm(5*pdim), ncol=pdim)

# run the code
out.small <- CovEst.2010RBLW(dat.small)

# visualize
opar <- par(mfrow=c(1,3), pty="s")
image(diag(pdim)[,pdim:1], main="true cov")
image(cov(dat.small)[,pdim:1], main="sample cov")
image(out.small$S[,,pdim:1], main="estimated cov")
```

```r
## Not run:
## want to see how delta is determined according to
## the number of observations we have.
nsamples = seq(from=5, to=200, by=5)
nnsample = length(nsamples)

# we will record two values; rho and norm difference
vec.rho = rep(0, nnsample)
vec.normd = rep(0, nnsample)
for (i in 1:nnsample){
dat.norun <- matrix(rnorm(nsamples[i]*pdim), ncol=pdim) # sample in R^5
dat.run <- CovEst.2010RBLW(dat.norun) # run with default
vec.rho[i] = dat.run$rho
vec.normd[i] = norm(dat.run$S - diag(5), "f") # Frobenius norm
}

# let's visualize the results
opar <- par(mfrow=c(1,2))
plot(nsamples, vec.rho, lwd=2, type="b", col="red", main="estimated rhos")
plot(nsamples, vec.normd, lwd=2, type="b", col="blue", main="Frobenius error")
par(opar)
```

## End(Not run)
Description

Cai and Liu (2011) proposed an adaptive variant of Bickel and Levina (2008) - CovEst.hard. The idea of adaptive thresholding is to apply thresholding technique on correlation matrix in that it becomes adaptive in terms of each variable.

Usage

CovEst.adaptive(X, thr = 0.5, nCV = 10, parallel = FALSE)

Arguments

X an \((n \times p)\) matrix where each row is an observation.

thr user-defined threshold value. If it is a vector of regularization values, it automatically selects one that minimizes cross validation risk.

nCV the number of repetitions for 2-fold random cross validations for each threshold value.

parallel a logical; TRUE to use half of available cores, FALSE to do every computation sequentially.

Value

a named list containing:

S a \((p \times p)\) covariance matrix estimate.

CV a dataframe containing vector of tested threshold values(thr) and corresponding cross validation scores(CVscore).

References


Examples

## generate data from multivariate normal with Identity covariance.
pdim <- 5
data <- matrix(rnorm(10*pdim), ncol=pdim)

## apply 4 different schemes
# mthr is a vector of regularization parameters to be tested
mthr <- seq(from=0.01,to=0.99,length.out=10)
```r
out1 <- CovEst.adaptive(data, thr=0.1) # threshold value 0.1
out2 <- CovEst.adaptive(data, thr=0.5) # threshold value 0.5
out3 <- CovEst.adaptive(data, thr=0.9) # threshold value 0.9
out4 <- CovEst.adaptive(data, thr=mthr) # automatic threshold checking

## visualize 4 estimated matrices
opar <- par(mfrow=c(2,2), pty="s")
image(out1$S[,pdim:1], col=gray((0:100)/100), main="thr=0.1")
image(out2$S[,pdim:1], col=gray((0:100)/100), main="thr=0.5")
image(out3$S[,pdim:1], col=gray((0:100)/100), main="thr=0.9")
image(out4$S[,pdim:1], col=gray((0:100)/100), main="automatic")
par(opar)
```

---

### CovEst.hard

#### Covariance Estimation via Hard Thresholding

**Description**

Bickel and Levina (2008) proposed a sparse covariance estimation technique to apply thresholding on off-diagonal elements of the sample covariance matrix. The entry of sample covariance matrix \( S_{i,j} = 0 \) if \(| S_{i,j} | \leq \tau\) where \( \tau \) is a thresholding value (\( thr \)). If \( thr \) is rather a vector of regularization parameters, it applies cross-validation scheme to select an optimal value.

**Usage**

```r
CovEst.hard(X, thr = sqrt(log(ncol(X))/nrow(X)), nCV = 10, parallel = FALSE)
```

**Arguments**

- \( X \) an \((n \times p)\) matrix where each row is an observation.
- \( thr \) user-defined threshold value. If it is a vector of regularization values, it automatically selects one that minimizes cross validation risk.
- \( nCV \) the number of repetitions for 2-fold random cross validations for each threshold value.
- \( parallel \) a logical; TRUE to use half of available cores, FALSE to do every computation sequentially.

**Value**

a named list containing:

- \( S \) a \((p \times p)\) covariance matrix estimate.
- \( CV \) a dataframe containing vector of tested threshold values(\( thr \)) and corresponding cross validation scores(\( CV_{score} \)).
References


Examples

```r
## generate data from multivariate normal with Identity covariance.
pdim <- 5
data <- matrix(rnorm(10*pdim), ncol=pdim)

## apply 4 different schemes
# mthr is a vector of regularization parameters to be tested
mthr <- exp(seq(from=log(0.1), to=log(10), length.out=10))
out1 <- CovEst.hard(data, thr=0.1)  # threshold value 0.1
out2 <- CovEst.hard(data, thr=1)    # threshold value 1
out3 <- CovEst.hard(data, thr=10)   # threshold value 10
out4 <- CovEst.hard(data, thr=mthr) # automatic threshold checking

## visualize 4 estimated matrices
gcol <- gray((0:100)/100)
opar <- par(mfrow=c(2,2), pty="s")
image(out1$S[,pdim:1], col=gcol, main="thr=0.1")
image(out2$S[,pdim:1], col=gcol, main="thr=1")
image(out3$S[,pdim:1], col=gcol, main="thr=10")
image(out4$S[,pdim:1], col=gcol, main="automatic")
par(opar)
```

---

**CovEst.hardPD**

*Covariance Estimation via Hard Thresholding under Positive-Definiteness Constraint*

**Description**

Sparse covariance estimation does not necessarily guarantee positive definiteness of an estimated covariance matrix. Fan et al. (2013) proposed to solve this issue by taking an iterative procedure to take an incremental decrease of threshold value until positive definiteness is preserved.

**Usage**

```r
CovEst.hardPD(X)
```

**Arguments**

- `X` an \((n \times p)\) matrix where each row is an observation.
CovEst.nearPD

Value

a named list containing:

S a \((p \times p)\) covariance matrix estimate.

optC an optimal threshold value \(C_{min}\) that guarantees positive definiteness after thresholding.

References


Examples

```r
## generate data from multivariate normal with Identity covariance.
pdim <- 5
data <- matrix(rnorm(10*pdim), ncol=pdim)

## apply 4 different schemes
out1 <- CovEst.hard(data, thr=0.1) # threshold value 0.1
out2 <- CovEst.hard(data, thr=1)  # threshold value 1
out3 <- CovEst.hard(data, thr=10) # threshold value 10
out4 <- CovEst.hardPD(data)  # automatic threshold checking

## visualize 4 estimated matrices
mmessage <- paste("hardPD::optimal thr=", sprintf("%.2f", out4$optC), sep="")
gcol <- gray((0:100)/100)
opar <- par(mfrow=c(2,2), pty="s")
image(out1$S[,pdim:1], col=gcol, main="thr=0.1")
image(out2$S[,pdim:1], col=gcol, main="thr=1")
image(out3$S[,pdim:1], col=gcol, main="thr=10")
image(out4$S[,pdim:1], col=gcol, main=mmessage)
par(opar)
```

---

CovEst.nearPD

Covariance Estimation via Nearest Positive-Definite Matrix Projection

Description

Qi and Sun (2006) proposed an algorithm for computing the positive correlation matrix with Positive Definiteness and transforming it back in order to estimate covariance matrix. This algorithm does not depend on any parameters.

Usage

CovEst.nearPD(X)
CovEst.soft

Covariance Estimation via Soft Thresholding

Description

Soft Thresholding method for covariance estimation takes off-diagonal elements \( z \) of sample covariance matrix and applies

\[
    h_\tau(z) = \text{sgn}(z)(|z| - \tau)_+
\]

where \( \text{sgn}(z) \) is a sign of the value \( z \), and \( (x)_+ = \max(x, 0) \). If \( \text{thr} \) is rather a vector of regularization parameters, it applies cross-validation scheme to select an optimal value.

Usage

\[
    \text{CovEst.soft}(X, \text{thr} = 0.5, \text{nCV} = 10, \text{parallel} = \text{FALSE})
\]
CovEst.soft

Arguments

X  an \((n \times p)\) matrix where each row is an observation.

thr  user-defined threshold value. If it is a vector of regularization values, it automatically selects one that minimizes cross validation risk.

nCV  the number of repetitions for 2-fold random cross validations for each threshold value.

parallel  a logical; TRUE to use half of available cores, FALSE to do every computation sequentially.

Value

a named list containing:

S  a \((p \times p)\) covariance matrix estimate.

CV  a dataframe containing vector of tested threshold values(thr) and corresponding cross validation scores(CVscore).

References


Examples

```r
## generate data from multivariate normal with Identity covariance.
pdim <- 5
data <- matrix(rnorm(10*pdim), ncol=pdim)

## apply 4 different schemes
# mthr is a vector of regularization parameters to be tested
mthr <- exp(seq(from=log(0.1),to=log(10),length.out=10))

out1 <- CovEst.soft(data, thr=0.1) # threshold value 0.1
out2 <- CovEst.soft(data, thr=1) # threshold value 1
out3 <- CovEst.soft(data, thr=10) # threshold value 10
out4 <- CovEst.soft(data, thr=mthr) # automatic threshold checking

## visualize 4 estimated matrices
# gcol <- gray((0:100)/100)
opar <- par(mfrow=c(2,2), pty="s")
image(out1$S[,pdim:1], col=gcol, main="thr=0.1")
image(out2$S[,pdim:1], col=gcol, main="thr=1")
image(out3$S[,pdim:1], col=gcol, main="thr=10")
image(out4$S[,pdim:1], col=gcol, main="automatic")
par(opar)
```
CovMean

*Estimate Mean Covariance Matrix*

**Description**

For a given 3-dimensional array where symmetric positive definite (SPD) matrices are stacked slice by slice, it estimates Frechet mean on an open cone of SPD matrices under corresponding metric/distance measure.

**Usage**

```r
CovMean(A, method = c("A", "C", "E", "L", "PS", "PF", "PE", "RE"), power = 1)
```

**Arguments**

- `A`: a `(p x p x N)` 3d array of `N` SPD matrices.
- `method`: the type of distance measures to be used;
  - "A" (AIRM) Affine Invariant Riemannian Metric
  - "C" (Cholesky) Cholesky difference in Frobenius norm
  - "E" (Euclidean) naive Frobenius norm as distance
  - "L" (LERM) Log Euclidean Riemannian Metric
  - "PS" (Procrustes.SS) Procrustes Size and Shape measure
  - "PF" (Procrustes.Full) Procrustes analysis with scale
  - "PE" (PowerEuclidean) weighted eigenvalues by some exponent
  - "RE" (RootEuclidean) matrix square root
- `power`: a non-zero number for PowerEuclidean distance.

**Value**

A `(p x p)` mean covariance matrix estimated.

**References**


**Examples**

```r
## Not run:
## generate 50 sample covariances of size (10-by-10).
pdim = 10
samples = samplecovs(50,pdim)

## compute means of first 50 sample covariances from data under Normal(0,Identity).
mA = CovMean(samples, method="A")
```
CovTest1.2013Cai

Description

Given data, it performs 1-sample test for Covariance where the null hypothesis is

\[ H_0 : \Sigma_n = \Sigma_0 \]

where \( \Sigma_n \) is the covariance of data model and \( \Sigma_0 \) is a hypothesized covariance based on a procedure proposed by Cai and Ma (2013).

Usage

\[
\text{CovTest1.2013Cai(data, Sigma0 = diag(ncol(data)), alpha = 0.05)}
\]

Arguments

data an \((n \times p)\) data matrix where each row is an observation.
Sigma0 a \((p \times p)\) given covariance matrix.
alpha level of significance.
Value

a named list containing:

- **statistic**: a test statistic value.
- **threshold**: rejection criterion to be compared against test statistic.
- **reject**: a logical; `TRUE` to reject null hypothesis, `FALSE` otherwise.

References


Examples

```r
## Not run:
## generate data from multivariate normal with trivial covariance.
pdim = 5
data = matrix(rnorm(10*pdim), ncol=pdim)

## run the test
CovTest1.2013Cai(data)

## End(Not run)
```

Description

Given data, it performs 1-sample test for Covariance where the null hypothesis is

\[ H_0 : \Sigma_n = \Sigma_0 \]

where \( \Sigma_n \) is the covariance of data model and \( \Sigma_0 \) is a hypothesized covariance based on a procedure proposed by Srivastava, Yanagihara, and Kubokawa (2014).

Usage

```
CovTest1.2014Srivastava(data, Sigma0 = diag(ncol(data)), alpha = 0.05)
```

Arguments

- **data**: an \((n \times p)\) data matrix where each row is an observation.
- **Sigma0**: a \((p \times p)\) given covariance matrix.
- **alpha**: level of significance.
Value

- a named list containing
  - **statistic** a test statistic value.
  - **threshold** rejection criterion to be compared against test statistic.
  - **reject** a logical; TRUE to reject null hypothesis, FALSE otherwise.

References


Examples

```r
## Not run:
## generate data from multivariate normal with trivial covariance.
pdim = 5
data = matrix(rnorm(10*pdim), ncol=pdim)
## run the test
CovTest1.2014Srivastava(data)
## End(Not run)
```

---

Two-Sample Covariance Test by Cai and Ma (2013)

Description

Given two sets of data, it performs 2-sample test for equality of covariance matrices where the null hypothesis is

\[ H_0 : \Sigma_1 = \Sigma_2 \]

where \( \Sigma_1 \) and \( \Sigma_2 \) represent true (unknown) covariance for each dataset based on a procedure proposed by Cai and Ma (2013). If statistic > threshold, it rejects null hypothesis.

Usage

CovTest2.2013Cai(X, Y, alpha = 0.05)

Arguments

- **X** an \((m \times p)\) matrix where each row is an observation from the first dataset.
- **Y** an \((n \times p)\) matrix where each row is an observation from the second dataset.
- **alpha** level of significance.
Value

a named list containing

**statistic** a test statistic value.

**threshold** rejection criterion to be compared against test statistic.

**reject** a logical; TRUE to reject null hypothesis, FALSE otherwise.

References

Cai TT, Ma Z (2013). “Optimal hypothesis testing for high dimensional covariance matrices.” *Bernoulli*, 19(5B), 2359–2388. ISSN 1350-7265, doi: [10.3150/12BEJ455](https://doi.org/10.3150/12BEJ455).

Examples

```r
## generate 2 datasets from multivariate normal with identical covariance.
pdim = 5
data1 = matrix(rnorm(100*pdim), ncol=pdim)
data2 = matrix(rnorm(150*pdim), ncol=pdim)

## run test
CovTest2.2013Cai(data1, data2)
```

Description

Given data, it performs 1-sample test for diagonal entries of a Covariance matrix where the null hypothesis is

\[ H_0 : \sigma_{ij} = 0 \text{ for any } i \neq j \]

and alternative hypothesis is \( H_1 : \text{not } H_0 \) with \( \Sigma_n = (\sigma_{ij}) \) based on a procedure proposed by Cai and Jiang (2011).

Usage

```r
DiagTest1.2011Cai(data, alpha = 0.05)
```

Arguments

- **data**: an \((n \times p)\) data matrix where each row is an observation.
- **alpha**: level of significance.
Value

a named list containing:

statistic  a test statistic value.

threshold  rejection criterion to be compared against test statistic.

reject    a logical; TRUE to reject null hypothesis, FALSE otherwise.

References


Examples

```r
## Not run:
## generate data from multivariate normal with trivial covariance.
pdim = 5
data = matrix(rnorm(100*pdim), ncol=pdim)

## run test with different alpha values
DiagTest1.2011Cai(data, alpha=0.01)
DiagTest1.2011Cai(data, alpha=0.05)
DiagTest1.2011Cai(data, alpha=0.10)

## End(Not run)
```

Description

Given data, it performs 1-sample test for diagonal entries of a Covariance matrix where the null hypothesis is

\[ H_0 : \sigma_{ij} = 0 \text{ for any } i \neq j \]

and alternative hypothesis is \( H_1 : \text{not } H_0 \) with \( \Sigma_n = (\sigma_{ij}) \) based on a procedure proposed by Lan et al. (2015).

Usage

```r
DiagTest1.2015Lan(data, alpha = 0.05)
```

Arguments

- **data** an \((n \times p)\) data matrix where each row is an observation.
- **alpha** level of significance.
Value

a named list containing:

- **statistic** a test statistic value.
- **threshold** rejection criterion to be compared against test statistic.
- **reject** a logical; TRUE to reject null hypothesis, FALSE otherwise.

References


Examples

```r
## Not run:
## generate data from multivariate normal with trivial covariance.
pdim = 5
data = matrix(rnorm(100*pdim), ncol=pdim)

## run test with different alpha values
DiagTest1.2015Lan(data, alpha=0.01)
DiagTest1.2015Lan(data, alpha=0.05)
DiagTest1.2015Lan(data, alpha=0.10)

## End(Not run)
```

package-CovTools

A Collection of Geometric and Statistical Tools for Covariance (and Precision) Analysis

Description

Covariance is of universal prevalence across various disciplines within statistics. **CovTools** package aims at providing a rich collection of geometric and statistical tools for a variety of inferences on covariance structures as well as its inverse called precision matrix. See the sections below for a comprehensive list of functions provided from the package.

Geometric Methods

From inference on manifolds perspective, we have following functions,

<table>
<thead>
<tr>
<th>name of a function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CovDist</td>
<td>compute pairwise distance of covariance matrices</td>
</tr>
<tr>
<td>CovMean</td>
<td>compute mean covariance matrix</td>
</tr>
</tbody>
</table>
**Banded Precision Matrix Estimation via Bandwidth Test**

**Description**

`PreEst.2014An` returns an estimator of the banded precision matrix using the modified Cholesky decomposition. It uses the estimator defined in Bickel and Levina (2008). The bandwidth is determined by the bandwidth test suggested by An, Guo and Liu (2014).

**Usage**

```r
PreEst.2014An(
  X, 
  upperK = floor(ncol(X)/2),
  algorithm = c("Bonferroni", "Holm"),
  alpha = 0.01
)
```

**Arguments**

- **X** an \((n \times p)\) data matrix where each row is an observation.
- **upperK** upper bound of bandwidth \(k\).
- **algorithm** bandwidth test algorithm to be used.
- **alpha** significance level for the bandwidth test.

**Value**

a named list containing:

- **C** a \((p \times p)\) estimated banded precision matrix.
- **optk** an estimated optimal bandwidth acquired from the test procedure.

**References**


**Examples**

```r
## Not run:
## parameter setting
p = 200; n = 100
k0 = 5; A0min=0.1; A0max=0.2; D0min=2; D0max=5
```
```
set.seed(123)
A0 = matrix(0, p,p)
for(i in 2:p){
    term1 = runif(n=min(k0,i-1),min=A0min, max=A0max)
    term2 = sample(c(1,-1),size=min(k0,i-1),replace=TRUE)
    vals = term1*term2
    vals = vals[ order(abs(vals)) ]
    A0[i, max(1, i-k0):(i-1)] = vals
}
D0 = diag(runif(n = p, min = D0min, max = D0max))
Omega0 = t(diag(p) - A0)

## data generation (based on AR representation)
## it is same with generating n random samples from N_p(0, Omega0^{-1})
X = matrix(0, nrow=n, ncol=p)
X[,1] = rnorm(n, sd = sqrt(D0[1,1]))
for(j in 2:p){
    mean.vec.j = X[, 1:(j-1)]
    X[,j] = rnorm(n, mean = mean.vec.j, sd = sqrt(D0[j,j]))
}

## banded estimation using two different schemes
Omega1 <- PreEst.2014An(X, upperK=20, algorithm="Bonferroni")
Omega2 <- PreEst.2014An(X, upperK=20, algorithm="Holm")

## visualize true and estimated precision matrices
opar <- par(mfrow=c(1,3), pty="s")
image(Omega0[,p:1], main="Original Precision")
image(Omega1$C[,p:1], main="banded3::Bonferroni")
image(Omega2$C[,p:1], main="banded3::Holm")
par(opar)

## End(Not run)
```
delta = 10,
logpi = function(k) { -k^4 },
loss = c("Stein", "Squared")
)

Arguments

X an \((n \times p)\) data matrix where each row is an observation.
upperK upper bound of bandwidth \(k\).
delta hyperparameter for G-Wishart prior. Default value is 10. It has to be larger than 2.
logpi log of prior distribution for bandwidth \(k\). Default is a function proportional to \(-k^4\).
loss type of loss; either "Stein" or "Squared".

Value

a named list containing:

\(C\) a \((p \times p)\) MAP estimate for precision matrix.

References


Examples

```r
## generate data from multivariate normal with Identity precision.
pdim = 10
data = matrix(rnorm(50*pdim), ncol=pdim)

## compare different K
out1 <- PreEst.2014Banerjee(data, upperK=1)
out2 <- PreEst.2014Banerjee(data, upperK=3)
out3 <- PreEst.2014Banerjee(data, upperK=5)

## visualize
opar <- par(mfrow=c(2,2), pty="s")
image(diag(pdim)[,pdim:1],main="Original Precision")
image(out1$C[,pdim:1], main="banded::upperK=1")
image(out2$C[,pdim:1], main="banded::upperK=3")
image(out3$C[,pdim:1], main="banded::upperK=5")
par(opar)
```
PreEst.2017Lee returns a Bayes estimator of the banded precision matrix, which is defined in subsection 3.3 of Lee and Lee (2017), using the k-BC prior. The bandwidth is set at the mode of marginal posterior for the bandwidth parameter.

Usage

PreEst.2017Lee(X, upperK = floor(ncol(X)/2), logpi = function(k) { -k^4 })

Arguments

X an \((n \times p)\) data matrix where each row is an observation.
upperK upper bound of bandwidth \(k\).
logpi log of prior distribution for bandwidth \(k\). Default is a function proportional to \(-k^4\).

Value

a named list containing:

\(C\) a \((p \times p)\) MAP estimate for precision matrix.

References


Examples

```r
## generate data from multivariate normal with Identity precision.
pdim = 5
data = matrix(rnorm(100*pdim), ncol=pdim)

## compare different K
out1 <- PreEst.2017Lee(data, upperK=1)
out2 <- PreEst.2017Lee(data, upperK=3)
out3 <- PreEst.2017Lee(data, upperK=5)

## visualize
opar <- par(mfrow=c(2,2), pty="s")
image(diag(pdim)[,pdim:1], main="Original Precision")
image(out1$C[,pdim:1], main="banded2::upperK=1")
```
Description

Given a sample covariance matrix $S$, graphical lasso aims at estimating sparse precision matrix $X$ - inverse of covariance. It solves a following optimization problem,

$$\max_X \log \det X - <S, X> - \lambda \|X\|_1 \text{ such that } X \succ 0$$

where $\lambda$ a regularization parameter, $<S, X> = tr(S^T X)$, $\|X\|_1 = \sum X_{ij}$ and $X \succ 0$ indicates positive definiteness. We provide three modes of computations, 'fixed', 'confidence', or 'BIC' with respect to $\lambda$. Please see the section below for more details.

Usage

PreEst.glasso(X, method = list(type = "fixed", param = 1), parallel = FALSE)

Arguments

- **X** an $(n \times p)$ data matrix where each row is an observation.
- **method** a list containing following parameters,
  - **type** one of 'fixed', 'confidence', or 'BIC'.
  - **param** either a numeric value or vector of values.
- **parallel** a logical; TRUE for using half the cores available, FALSE otherwise.

Value

a named list containing:

- **C** a $(p \times p)$ estimated precision matrix.
- **BIC** a dataframe containing $\lambda$ values and corresponding BIC scores with type='BIC' method.

regularization parameters

We currently provide three options for solving the problem, 'fixed', 'confidence', or 'BIC' with respect to $\lambda$. When the method type is 'fixed', the parameter should be a single numeric value as a user-defined $\lambda$ value. Likewise, method type of 'confidence' requires a single numeric value in $(0, 1)$, where the value is set heuristically according to

$$\rho = \frac{t_{n-2}(\gamma) \max S_{ii} S_{jj}}{\sqrt{n - 2 + t_{n-2}^2(\gamma)}}$$

for a given confidence level $\gamma \in (0, 1)$ as proposed by Banerjee et al. (2006). Finally, 'BIC' type requires a vector of $\lambda$ values and opts for a lambda value with the lowest BIC values as proposed by Yuan and Lin (2007).
References


Examples

```r
## generate data from multivariate normal with Identity precision.
pdim = 10
data = matrix(rnorm(100*pdim), ncol=pdim)

## prepare input arguments for different scenarios
lbdvec <- c(0.01,0.1,1,10,100) # a vector of regularization parameters
list1 <- list(type="fixed",param=1.0) # single regularization parameter case
list2 <- list(type="confidence",param=0.95) # single confidence level case
list3 <- list(type="BIC",param=lbdvec) # multiple regularizers with BIC selection

## compute with different scenarios
out1 <- PreEst.glasso(data, method=list1)
out2 <- PreEst.glasso(data, method=list2)
out3 <- PreEst.glasso(data, method=list3)

## visualize
opar <- par(mfrow=c(2,2), pty="s")
image(diag(pdim)[,pdim:1], main="Original Precision")
image(out1$C[,pdim:1], main="glasso::lambda=1.0")
image(out2$C[,pdim:1], main="glasso::Confidence=0.95")
image(out3$C[,pdim:1], main="glasso::BIC selection")
par(opar)
```

samplecovs

*Generate Sample Covariances of 2 groups*

**Description**

For visualization purpose, samplecovs generates a 3d array of stacked sample covariances where - in 3rd dimension, the first half are sample covariances of samples generated independently from normal distribution with identity covariance, where the latter half consists of samples covariances from dense random population covariance.
Usage

```r
samplecovs(ncopy, size)
```

Arguments

- `ncopy`: the total number of sample covariances to be generated.
- `size`: dimension \( p \).

Value

A \( (p \times p \times ncopy) \) array of strictly positive definite sample covariances.

Examples

```r
## generate total of 20 samples covariances of size 5-by-5.
samples <- samplecovs(20, 5)
```
Index

BCovTest1.mxPBF, 2
BDiagTest1.mxPBF, 4

CovDist, 5, 24
CovEst.2003LW, 7
CovEst.2010OAS, 8
CovEst.2010RBLW, 10
CovEst.adaptive, 12
CovEst.hard, 12, 13
CovEst.hardPD, 14
CovEst.nearPD, 15
CovEst.soft, 16
CovMean, 18, 24
CovTest1.2013Cai, 19
CovTest1.2014Srivastava, 20
CovTest2.2013Cai, 21

DiagTest1.2011Cai, 22
DiagTest1.2015Lan, 23

package-CovTools, 24
PreEst.2014An, 25
PreEst.2014Banerjee, 26
PreEst.2017Lee, 28
PreEst.glasso, 29

samplecocs, 30