Package 'CovTools'

January 29, 2019

Type Package

Title Statistical Tools for Covariance Analysis

Version 0.5.1

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

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.1.1

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 <kyou@nd.edu>

Repository CRAN

Date/Publication 2019-01-29 09:00:07 UTC

R topics documented:

BCovTest1.mxPBF 2
BDiagTest1.mxPBF 3

CovDist 5
CovEst.adaptive 6
CovEst.hard 7
CovEst.hardPD 8
CovEst.nearPD 9
CovEst.soft 10
CovMean 12
CovTest1.2013Cai 13
CovTest1.2014Srivastava 14
CovTest2.2013Cai 15
DiagTest1.2011Cai 16
DiagTest1.2015Lan 17
package-CovTools 18
PreEst.2014An 19
PreEst.2014Banerjee 20
PreEst.2017Lee 22
PreEst.glasso 23
samplecovs 24

Index **26**

`BCovTest1.mxPBF` *One-Sample Covariance Test using Maximum Pairwise Bayes Factor*

Description

It performs Bayesian version of 1-sample test for Covariance where the null hypothesis is

$$H_0 : \Sigma_n = \Sigma_0$$

where Σ_n is the covariance of data model and Σ_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,

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

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

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

$$\tau_{ij}^2 \sim IG(a0, b0).$$

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

Lee K, Lin L, Dunson D (2018). "Maximum Pairwise Bayes Factors for Covariance Structure Testing." *arXiv:1809.03105 [stat]*. arXiv: 1809.03105, http://arxiv.org/abs/1809.03105.

Examples

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

## run mxPBF-based test
out1 = BCovTest1.mxPBF(data)
out2 = BCovTest1.mxPBF(data, a0=5.0, b0=5.0) # change some params

## visualize two Bayes Factor matrices
par(mfrow=c(1,2), pty="s")
image(exp(out1$log.BF.mat)[,5:1], main="default")
image(exp(out2$log.BF.mat)[,5:1], main="a0=b0=5.0")

## End(Not run)
```

`BDiagTest1.mxPBF` *One-Sample Diagonality Test by Maximum Pairwise Bayes Factor*

Description

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

$$H_0 : \sigma_{ij} = 0 \text{ forany } i \neq j$$

and alternative hypothesis $H_1 :$ 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,

$$H_0 : a_{ij} = 0 \quad \text{versus.} \quad H_1 : \text{not } H_0.$$

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

$$\tau_{ij}^2 \sim IG(a0, b0)$$

and under H_1, priors are

$$a_{ij}|\tau_{ij}^2 \sim N(0, \tau_{ij}^2/(\gamma * ||X_j||^2))$$
$$\tau_{ij}^2 \sim IG(a0, b0).$$

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

a named list containing:

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

References

Lee K, Lin L, Dunson D (2018). “Maximum Pairwise Bayes Factors for Covariance Structure Testing.” *arXiv:1809.03105 [stat]*. arXiv: 1809.03105, http://arxiv.org/abs/1809.03105.

Examples

```
## Not run:
## generate data from multivariate normal with trivial covariance.
data = matrix(rnorm(100*5), 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
par(mfrow=c(1,2), pty="s")
image(exp(out1$log.BF.mat)[,5:1], main="default")
image(exp(out2$log.BF.mat)[,5:1], main="a0=b0=5.0")

## 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

```
CovDist(A, method = c("AIRM", "Bhattacharyya", "Cholesky", "Euclidean",
  "Hellinger", "JBLD", "KLDM", "LERM", "Procrustes.SS", "Procrustes.Full",
  "PowerEuclidean", "RootEuclidean"), power = 1)
```

Arguments

`A`	a $(p \times p \times N)$ 3d array of N SPD matrices.
`method`	the type of distance measures to be used; `"AIRM"` for Affine Invariant Riemannian Metric, `"Bhattacharyya"` for Bhattacharyya distance based on normal model, `"Cholesky"` for Cholesky difference in Frobenius norm, `"Euclidean"` for naive Frobenius norm as distance, `"Hellinger"` for Hellinger distance based on normal model, `"JBLD"` for Jensen-Bregman Log Determinant Distance, `"KLDM"` for symmetrized Kullback-Leibler Distance Measure, `"LERM"` for Log Euclidean Riemannian Metric, `"Procrustes.SS"` for Procrustes Size and Shape measure, `"Procrustes.Full"` for Procrustes analysis with scale, `"PowerEuclidean"` for weighted eigenvalues by some exponent, and `"RootEuclidean"` for matrix square root.
`power`	a non-zero number for PowerEuclidean distance.

Value

an $(N \times N)$ symmetric matrix of pairwise distances.

References

Arsigny V, Fillard P, Pennec X, Ayache N (2006). "Log-Euclidean metrics for fast and simple calculus on diffusion tensors." *Magnetic Resonance in Medicine*, **56**(2), 411–421. ISSN 0740-3194, 1522-2594, doi: 10.1002/mrm.20965, `http://doi.wiley.com/10.1002/mrm.20965`.

Dryden IL, Koloydenko A, Zhou D (2009). "Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging." *The Annals of Applied Statistics*, **3**(3), 1102–1123. ISSN 1932-6157, doi: 10.1214/09AOAS249, `2017-12-03`.

Examples

```
## generate 100 SPD matrices of size (5-by-5)
samples = samplecovs(100,5)

## get pairwise distance for "AIRM"
distAIRM = CovDist(samples, method="AIRM")

## dimension reduction using MDS
ss = cmdscale(distAIRM)
plot(ss[,1],ss[,2],main="2d projection")
```

`CovEst.adaptive` *Covariance Estimation via Adaptive Thresholding*

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

Cai T, Liu W (2011). “Adaptive Thresholding for Sparse Covariance Matrix Estimation.” *Journal of the American Statistical Association*, **106**(494), 672–684. ISSN 0162-1459, 1537-274X, doi: 10.1198/jasa.2011.tm10560, `2017-12-12`.

Examples

```
## generate data from multivariate normal with Identity covariance.
data <- mvtnorm::rmvnorm(10, sigma=diag(10))

## 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)

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=0.5)  # threshold value 0.9
out4 <- CovEst.adaptive(data, thr=mthr) # automatic threshold checking

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

`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 $|S_{i,j}| <= \tau$ where τ 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

```
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(`CVscore`).

References

Bickel PJ, Levina E (2008). "Covariance regularization by thresholding." *The Annals of Statistics*, **36**(6), 2577–2604. ISSN 0090-5364, doi: 10.1214/08AOS600, `2017-12-01`.

Examples

```
## generate data from multivariate normal with Identity covariance.
data <- mvtnorm::rmvnorm(10, sigma=diag(10))

## 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
par(mfrow=c(2,2), pty="s")
image(pracma::flipud(out1$S), col=gray((0:100)/100), main="thr=0.1")
image(pracma::flipud(out2$S), col=gray((0:100)/100), main="thr=1")
image(pracma::flipud(out3$S), col=gray((0:100)/100), main="thr=10")
image(pracma::flipud(out4$S), col=gray((0:100)/100), main="automatic")
```

`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

```
CovEst.hardPD(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.

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

References

Fan J, Liao Y, Mincheva M (2013). "Large covariance estimation by thresholding principal orthogonal complements." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **75**(4), 603–680. ISSN 13697412, doi: 10.1111/rssb.12016, `2018-02-12`.

Examples

```
## generate data from multivariate normal with Identity covariance.
data <- mvtnorm::rmvnorm(5, sigma=diag(10))

## 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$optC),sep="")
par(mfrow=c(2,2), pty="s")
image(pracma::flipud(out1$S), col=gray((0:100)/100), main="thr=0.1")
image(pracma::flipud(out2$S), col=gray((0:100)/100), main="thr=1")
image(pracma::flipud(out3$S), col=gray((0:100)/100), main="thr=10")
image(pracma::flipud(out4$S), col=gray((0:100)/100), main=mmessage)
```

`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)
```

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.

References

Qi H, Sun D (2006). "A Quadratically Convergent Newton Method for Computing the Nearest Correlation Matrix." *SIAM Journal on Matrix Analysis and Applications*, **28**(2), 360–385. ISSN 0895-4798, 1095-7162, doi: 10.1137/050624509, `2018-02-12`.

Examples

```
## generate data from multivariate normal with Identity covariance.
data <- mvtnorm::rmvnorm(3, sigma=diag(10))

## compare against sample covariance
out1 <- cov(data)
out2 <- CovEst.nearPD(data) # apply nearPD

## visualize 2 estimated matrices
par(mfrow=c(1,2), pty="s")
image(pracma::flipud(out1), col=gray((0:100)/100), main="sample covariance")
image(pracma::flipud(out2$S), col=gray((0:100)/100), main="SPD Projection")
```

`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 `thr` is rather a vector of regularization parameters, it applies cross-validation scheme to select an optimal value.

Usage

```
CovEst.soft(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

Antoniadis A, Fan J (2001). “Regularization of Wavelet Approximations.” *Journal of the American Statistical Association*, **96**(455), 939–967. ISSN 0162-1459, 1537-274X, doi: 10.1198/016214501753208942, `2018-02-11`.

Donoho DL, Johnstone IM, Kerkyacharian G, Picard D (1995). “Wavelet Shrinkage: Asymptopia?” *Journal of the Royal Statistical Society. Series B (Methodological)*, **57**(2), 301–369. ISSN 00359246.

Examples

```
## generate data from multivariate normal with Identity covariance.
data <- mvtnorm::rmvnorm(10, sigma=diag(10))

## 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
par(mfrow=c(2,2), pty="s")
image(pracma::flipud(out1$S), col=gray((0:100)/100), main="thr=0.1")
image(pracma::flipud(out2$S), col=gray((0:100)/100), main="thr=1")
image(pracma::flipud(out3$S), col=gray((0:100)/100), main="thr=10")
image(pracma::flipud(out4$S), col=gray((0:100)/100), main="automatic")
```

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

```
CovMean(A, method = c("AIRM", "Cholesky", "Euclidean", "LERM",
  "Procrustes.SS", "Procrustes.Full", "PowerEuclidean", "RootEuclidean"),
  power = 1)
```

Arguments

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

Value

a $(p \times p)$ mean covariance matrix estimated.

References

Dryden IL, Koloydenko A, Zhou D (2009). “Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging.” *The Annals of Applied Statistics*, **3**(3), 1102–1123. ISSN 1932-6157, doi: 10.1214/09AOAS249, 2017-12-03.

Examples

```
## Not run:
## generate 100 sample covariances of size (5-by-5).
samples = samplecovs(100,5)

## Compute mean of first 50 sample covariances from data under Normal(0,Identity).
mLERM = CovMean(samples[,,1:50],method="LERM")
mAIRM = CovMean(samples[,,1:50],method="AIRM")

## End(Not run)
```

`CovTest1.2013Cai` *One-Sample Covariance Test by Cai and Ma (2013)*

Description

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

$$H_0 : \Sigma_n = \Sigma_0$$

where Σ_n is the covariance of data model and Σ_0 is a hypothesized covariance based on a procedure proposed by Cai and Ma (2013).

Usage

```
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

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, `2018-02-19`.

Examples

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

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

## End(Not run)
```

`CovTest1.2014Srivastava` *One-Sample Covariance Test by Srivastava, Yanagihara, and Kubokawa (2014)*

Description

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

$$H_0 : \Sigma_n = \Sigma_0$$

where Σ_n is the covariance of data model and Σ_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

Srivastava MS, Yanagihara H, Kubokawa T (2014). "Tests for covariance matrices in high dimension with less sample size." *Journal of Multivariate Analysis*, **130**, 289–309. ISSN 0047259X, doi: 10.1016/j.jmva.2014.06.003, `2018-09-14`.

Examples

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

## run the test
CovTest1.2014Srivastava(data)

## End(Not run)
```

`CovTest2.2013Cai` *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 Σ_1 and Σ_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, `2018-02-19`.

Examples

```
## generate 2 datasets from multivariate normal with identical covariance.
data1 = mvtnorm::rmvnorm(100, sigma=diag(5))
data2 = mvtnorm::rmvnorm(200, sigma=diag(5))

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

`DiagTest1.2011Cai` *One-Sample Diagonality Test by Cai and Jiang (2011)*

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

```
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

Cai TT, Jiang T (2011). "Limiting laws of coherence of random matrices with applications to testing covariance structure and construction of compressed sensing matrices." *The Annals of Statistics*, **39**(3), 1496–1525. ISSN 0090-5364, doi: 10.1214/11AOS879, `http://projecteuclid.org/euclid.aos/1305292044`.

Examples

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

## 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)
```

`DiagTest1.2015Lan` *One-Sample Diagonality Test by Lan et al. (2015)*

Description

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

$$H_0 : \sigma_{ij} = 0 \text{ forany } i \neq j$$

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

Usage

```
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

Lan W, Luo R, Tsai C, Wang H, Yang Y (2015). "Testing the Diagonality of a Large Covariance Matrix in a Regression Setting." *Journal of Business \& Economic Statistics*, **33**(1), 76–86. ISSN 0735-0015, 1537-2707, doi: 10.1080/07350015.2014.923317, `http://www.tandfonline.com/doi/abs/10.1080/07350015.2014.923317`.

Examples

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

## 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,

name of a function	*description*
`CovDist`	compute pairwise distance of covariance matrices
`CovMean`	compute mean covariance matrix

Statistical methods

We provide statistical methods for (1) **Covariance Matrix Estimation**,

name of a function	*description*
`CovEst.adaptive`	Adaptive Thresholding.
`CovEst.hard`	Hard Thresholding.
`CovEst.hardPD`	Hard Thresholding under Positive-Definiteness Constraint.
`CovEst.nearPD`	Nearest Positive-Definite Matrix Projection.
`CovEst.soft`	Soft Thresholding.

(2) **Precision Matrix Estimation**

name of a function	*description*
`PreEst.2014An`	Banded Precision Matrix Estimation via Bandwidth Test.
`PreEst.2014Banerjee`	Bayesian Estimation of a Banded Precision Matrix (Banerjee 2014).
`PreEst.2017Lee`	Bayesian Estimation of a Banded Precision Matrix (Lee 2017).
`PreEst.glasso`	Graphical Lasso.

(3) **1-Sample Covariance Tests**

name of a function	*description*
`BCovTest1.mxPBF`	Bayesian Test using Maximum Pairwise Bayes Factor.
`CovTest1.2013Cai`	Test by Cai and Ma (2013).
`CovTest1.2014Srivastava`	Test by Srivastava, Yanagihara, and Kubokawa (2014).

(4) **2-Sample Covariance Tests**

name of a function	*description*
`CovTest2.2013Cai`	Test by Cai and Ma (2013).

(5) **1-Sample Diagonality Tests**

name of a function	*description*
`BDiagTest1.mxPBF`	Bayesian Test using Maximum Pairwise Bayes Factor.
`DiagTest1.2011Cai`	Test by Cai and Jiang (2011).
`DiagTest1.2015Lan`	Test by Lan et al. (2015).

`PreEst.2014An` *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

```
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

An B, Guo J, Liu Y (2014). “Hypothesis testing for band size detection of high-dimensional banded precision matrices.” *Biometrika*, **101**(2), 477–483. ISSN 0006-3444, 1464-3510, doi: 10.1093/biomet/asu006, `2018-02-18`.

Bickel PJ, Levina E (2008). “Regularized estimation of large covariance matrices.” *The Annals of Statistics*, **36**(1), 199–227. ISSN 0090-5364, doi: 10.1214/009053607000000758, `2018-02-18`.

Examples

```
## 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)%*%diag(1/diag(D0))%*%(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)]%*%as.matrix(A0[j, 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
par(mfrow=c(1,3), pty="s")
image(pracma::flipud(Omega0), main="Original Precision")
image(pracma::flipud(Omega1$C), main="banded3::Bonferroni")
image(pracma::flipud(Omega2$C), main="banded3::Holm")

## End(Not run)
```

`PreEst.2014Banerjee` *Bayesian Estimation of a Banded Precision Matrix (Banerjee 2014)*

Description

`PreEst.2014Banerjee` returns a Bayes estimator of the banded precision matrix using G-Wishart prior. Stein's loss or squared error loss function is used depending on the "loss" argument in the function. The bandwidth is set at the mode of marginal posterior for the bandwidth parameter.

Usage

```
PreEst.2014Banerjee(X, upperK = floor(ncol(X)/2), 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

Banerjee S, Ghosal S (2014). “Posterior convergence rates for estimating large precision matrices using graphical models.” *Electronic Journal of Statistics*, **8**(2), 2111–2137. ISSN 1935-7524, doi: 10.1214/14EJS945, `2018-02-13`.

Examples

```
## generate data from multivariate normal with Identity precision.
data = mvtnorm::rmvnorm(100, sigma=diag(10))

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

## visualize
par(mfrow=c(2,2), pty="s")
image(pracma::flipud(diag(10)),main="Original Precision")
image(pracma::flipud(out1$C), main="banded1::upperK=1")
image(pracma::flipud(out2$C), main="banded1::upperK=3")
image(pracma::flipud(out3$C), main="banded1::upperK=5")
```

PreEst.2017Lee *Bayesian Estimation of a Banded Precision Matrix (Lee 2017)*

Description

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

Lee K, Lee J (2017). "Estimating Large Precision Matrices via Modified Cholesky Decomposition." *ArXiv e-prints*.

Examples

```
## generate data from multivariate normal with Identity precision.
data = mvtnorm::rmvnorm(100, sigma=diag(10))

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

## visualize
par(mfrow=c(2,2), pty="s")
image(pracma::flipud(diag(10)),main="Original Precision")
image(pracma::flipud(out1$C), main="banded2::upperK=1")
image(pracma::flipud(out2$C), main="banded2::upperK=3")
image(pracma::flipud(out3$C), main="banded2::upperK=5")
```

`PreEst.glasso` *Precision Matrix Estimation via Graphical Lasso*

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 λ 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 λ. 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 λ 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 λ. When the method type is `'fixed'`, the parameter should be a single numeric value as a user-defined λ value. Likewise, method type of `'confidence'` requires a singule 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 λ values and opts for a lambda value with the lowest BIC values as proposed by Yuan and Lin (2007).

References

Banerjee O, Ghaoui LE, d'Aspremont A, Natsoulis G (2006). "Convex optimization techniques for fitting sparse Gaussian graphical models." In *Proceedings of the 23rd international conference on Machine learning*, 89–96. ISBN 978-1-59593-383-6, doi: 10.1145/1143844.1143856, 2018-02-12.

Yuan M, Lin Y (2007). "Model Selection and Estimation in the Gaussian Graphical Model." *Biometrika*, **94**(1), 19–35. ISSN 00063444.

Friedman J, Hastie T, Tibshirani R (2008). "Sparse inverse covariance estimation with the graphical lasso." *Biostatistics*, **9**(3), 432–441. ISSN 1465-4644, 1468-4357, doi: 10.1093/biostatistics/kxm045, 2018-02-12.

Examples

```
## generate data from multivariate normal with Identity precision.
data = mvtnorm::rmvnorm(100, sigma=diag(10))

## prepare input arguments for diefferent 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
par(mfrow=c(2,2), pty="s")
image(pracma::flipud(diag(10)),main="Original Precision")
image(pracma::flipud(out1$C), main="glasso::lambda=1.0")
image(pracma::flipud(out2$C), main="glasso::Confidence=0.95")
image(pracma::flipud(out3$C), main="glasso::BIC selection")
```

`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

```
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

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

Index

BCovTest1.mxPBF, 2, *18*
BDiagTest1.mxPBF, 3, *19*

CovDist, 5, *18*
CovEst.adaptive, 6, *18*
CovEst.hard, *6*, 7, *18*
CovEst.hardPD, 8, *18*
CovEst.nearPD, 9, *18*
CovEst.soft, 10, *18*
CovMean, 12, *18*
CovTest1.2013Cai, 13, *18*
CovTest1.2014Srivastava, 14, *18*
CovTest2.2013Cai, 15, *19*

DiagTest1.2011Cai, 16, *19*
DiagTest1.2015Lan, 17, *19*

package-CovTools, 18
package-CovTools-package
    (package-CovTools), 18
PreEst.2014An, *18*, 19
PreEst.2014Banerjee, *18*, 20
PreEst.2017Lee, *18*, 22
PreEst.glasso, *18*, 23

samplecovs, 24