Package ‘CovTools’

February 12, 2018

Type Package

Title Statistical Tools for Covariance Analysis

Version 0.2.1

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

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

RdMacros Rdpack

Author Kyoungjae Lee [aut], Lizhen Lin [ctb], Kisung You [aut, cre] (0000-0002-8584-459X)

Maintainer Kisung You <kyou@nd.edu>

R topics documented:

`CovTools-package` *A Collection of Geometric and Statistical Tools for Covariance Analysis*

Description

Covariance is of universal prevalence across various disciplines within statistics. The **CovTools** aims at providing a rich collection of geometric and statistical tools for a variety of inferences on covariance structures. Following is a list of functions,

name of a function	*description*
`CovDist`	compute pairwise distance of covariance matrices
`CovMean`	compute mean covariance matrix
`CovTest1`	1-sample tests for covariance matrix
`CovTest2`	2-sample tests for covariance matrices

Details

Also, below is a list of functions for *estimating* covariance matrices,

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

and precision - inverse covariance - matrices,

name of a function	*description*
`PreEst.glasso`	Graphical Lasso

`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-by-p-by-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-by-N)` symmetric matrix of pairwise distances.

References

Arsigny, V., Fillard, P., Pennec, X., and Ayache, N. (2006) *Log-Euclidean Metrics for Fast and Simple Calculus on Diffusion Tensors.* Magnetic Resonance in Medicine, 56:411-421.

Dryden, I.L., Koloydenko, A., and Zhou, D. (2009) *Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging.* The Annals of Applied Statistics, Vol.3(3):1102-1123.

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 and Liu W (2011). “Adaptive Thresholding for Sparse Covariance Matrix Estimation.” *Journal of the American Statistical Association*, **106**(494), pp. 672–684. ISSN 0162-1459, 1537-274X, doi: 10.1198/jasa.2011.tm10560, `http://www.tandfonline.com/doi/abs/10.1198/jasa.2011.tm10560`.

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 and Levina E (2008). "Covariance regularization by thresholding." *The Annals of Statistics*, **36**(6), pp. 2577–2604. ISSN 0090-5364, doi: 10.1214/08AOS600, `http://projecteuclid.org/euclid.aos/1231165180`.

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 and Mincheva M (2013). "Large covariance estimation by thresholding principal orthogonal complements." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **75**(4), pp. 603–680. ISSN 13697412, doi: 10.1111/rssb.12016, `http://doi.wiley.com/10.1111/rssb.12016`.

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

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

$$h_\tau(z) = \mathrm{sgn}(z)(|z| - \tau)_+$$

where $\mathrm{sgn}(z)$ 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 and Fan J (2001). "Regularization of Wavelet Approximations." *Journal of the American Statistical Association*, **96**(455), pp. 939–967. ISSN 0162-1459, 1537-274X, doi: 10.1198/016214501753208942, `http://www.tandfonline.com/doi/abs/10.1198/016214501753208942`.

Donoho DL, Johnstone IM, Kerkyacharian G and Picard D (1995). "Wavelet Shrinkage: Asymptopia?" *Journal of the Royal Statistical Society. Series B (Methodological)*, **57**(2), pp. 301–369. ISSN 00359246, `http://www.jstor.org/stable/2345967`.

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-by-p-by-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-by-p`) mean covariance matrix estimated.

References

Dryden, I.L., Koloydenko, A., and Zhou, D. (2009) *Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging.* The Annals of Applied Statistics, Vol.3(3):1102-1123.

Examples

```
## 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")
```

CovTest1 *One-Sample Tests for Covariance Matrices*

Description

Given data, `CovTest1` 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.

Usage

```
CovTest1(data, Sigma0 = diag(ncol(data)), alpha = 0.05,
  method = c("Cai13"))
```

Arguments

`data`	an `(n-by-p)` data matrix where each row is an observation.
`Sigma0`	a `(p-by-p)` given covariance matrix.
`alpha`	level of significance.
`method`	a name of test.

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

[Cai13] Cai, T. and Ma, Z. (2013) *Optimal hypothesis testing for high dimensional covariance matrices.* Bernoulli, Vol.19(5B):2359-2388.

Examples

```
## generate data from multivariate normal with trivial covariance.
data = mvtnorm::rmvnorm(100, sigma=diag(5))

## run test
CovTest1(data, Sigma0=diag(5))
```

CovTest2 *Two-Sample Tests for Covariance Matrices*

Description

Given two sets of data, `CovTest2` performs 2-sample test for Covariance where the null hypothesis is

$$H_0 : \Sigma_1 = \Sigma_2$$

where Σ_1 and Σ_2 represent true (unknown) covariance for each dataset.

Usage

```
CovTest2(X, Y, alpha = 0.05, method = c("Cai13"))
```

Arguments

`X`	an `(m-by-p)` matrix where each row is an observation from the first dataset.
`Y`	an `(n-by-p)` matrix where each row is an observation from the second dataset.
`alpha`	level of significance.
`method`	a name of test.

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

[Cai13] Cai, T., Liu, W., and Xia, Y. (2013) *Two-Sample Covariance Matrix Testing and Support Recovery in High-Dimensional and Sparse Settings.* Journal of American Statistical Association, Vol.108(501):265-277.

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(data1, data2)
```

PreEst *Estimating Precision Matrix*

Description

This code compiles several estimation methods for *precision* matrix, which is an inverse of covariance matrix, including penalized likelihood method with L1 penalty(`Yuan07`, `Banerjee06`) or Bayesian approaches incorporating banded structure assumptions(`Banerjee14`,`Lee17`).

Usage

```
PreEst(X, method = c("Banerjee06", "Banerjee14", "Lee17", "Yuan07"),
  opt = list(Banerjee06.confidence = 0.95, Banerjee14.upperK =
  floor(ncol(X)/2), Banerjee14.delta = 10, Banerjee14.logpi = function(k) {
  -k^4 }, Banerjee14.loss = c("Stein", "Squared"), Lee17.upperK =
  floor(ncol(X)/2), Lee17.logpi = function(k) {     -k^4 }, Yuan07.lambda = 1))
```

Arguments

`X`	an `(n-by-p)` matrix where each row is an observation from the first dataset.
`method`	a name of estimation method.
`opt`	a list containing following parameters,

- **Banerjee06.confidence** level of confidence in $(0, 1)$.
- **Banerjee14.upperK** upper bound for bandwidth.
- **Banerjee14.delta** a number larger than 2.
- **Banerjee14.logpi** log of prior distribution for bandwidth `k`.
- **Banerjee14.loss** loss type; either `"Stein"` or `"Squared"` type.
- **Lee17.upperK** upper bound for bandwidth.
- **Lee17.logpi** log of prior distribution for bandwidth `k`.
- **Yuan07.lambda** a regularization parameter.

Value

a `(p-by-p)` estimated precision matrix.

References

[Banerjee06] Banerjee et al (2006) *Convex optimization techniques for fitting sparse Gaussian graphical models.* ICML'06:89-96.

[Banerjee14] Banerjee, S. and Ghosal, S. (2014) *Posterior convergence rates for estimating large precision matrices using graphical models.* Electronic Journal of Statistics, Vol.8:2111-2137.

[Lee17] Lee, K. and Lee, J. (2017) *Estimating Large Precision Matrices via Modified Cholesky Decomposition.* arXiv:1707.01143.

[Yuan07] Yuan, M. and Lin, Y. (2007) *Model selection and estimation in the Gaussian graphical model.* Biometrika, Vol.94(1):19-35.

Examples

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

## run estimation
out1 = PreEst(data, method="Banerjee06")
out2 = PreEst(data, method="Banerjee14")
out3 = PreEst(data, method="Lee17")
out4 = PreEst(data, method="Yuan07")

## Visualize
par(mfrow=c(2,2))
image(pracma::flipud(out1), main="Banerjee06")
image(pracma::flipud(out2), main="Banerjee14")
image(pracma::flipud(out3), main="Lee17")
image(pracma::flipud(out4), main="Yuan07")
```

`PreEst.auto` *Estimating Precision Matrix with automatic parameter tuning*

Description

This code compiles several estimation methods for *precision* matrix, which is an inverse of covariance matrix, including penalized likelihood method with L1 penalty(`Yuan07`, `Banerjee06`) or Bayesian approaches incorporating banded structure assumptions(`Banerjee14`,`Lee17`). At this moment, the only automation is necessary for the method `"Yuan07"` for multiple trials with different λ values.

Usage

```
PreEst.auto(X, method = c("Banerjee06", "Banerjee14", "Lee17", "Yuan07"),
  opt = list(Banerjee06.confidence = 0.95, Banerjee14.upperK =
  floor(ncol(X)/2), Banerjee14.delta = 10, Banerjee14.logpi = function(k) {
  -k^4 }, Banerjee14.loss = c("Stein", "Squared"), Lee17.upperK =
  floor(ncol(X)/2), Lee17.logpi = function(k) {     -k^4 }, Yuan07.lambdagrid =
  seq(from = 0.01, to = 2, length.out = 10)), parallel = FALSE)
```

Arguments

`X`	an `(n-by-p)` matrix where each row is an observation from the first dataset.
`method`	a name of estimation method.
`opt`	a list containing following parameters, **Banerjee06.confidence** level of confidence in $(0, 1)$. **Banerjee14.upperK** upper bound for bandwidth. **Banerjee14.delta** a number larger than 2.

Banerjee14.logpi log of prior distribution for bandwidth `k`.

Banerjee14.loss loss type; either `"Stein"` or `"Squared"` type.

Lee17.upperK upper bound for bandwidth.

Lee17.logpi log of prior distribution for bandwidth `k`.

Yuan07.lambdagrid a vector of regularization parameters.

`parallel` a logical; `TRUE` for using half the cores available, `FALSE` otherwise.

Value

a named list containing:

C a `(p-by-p)` estimated precision matrix.

BIC a vector of BIC scores for the method `Yuan07`, only.

References

[Banerjee06] Banerjee et al (2006) *Convex optimization techniques for fitting sparse Gaussian graphical models.* ICML'06:89-96.

[Banerjee14] Banerjee, S. and Ghosal, S. (2014) *Posterior convergence rates for estimating large precision matrices using graphical models.* Electronic Journal of Statistics, Vol.8:2111-2137.

[Lee17] Lee, K. and Lee, J. (2017) *Estimating Large Precision Matrices via Modified Cholesky Decomposition.* arXiv:1707.01143.

[Yuan07] Yuan, M. and Lin, Y. (2007) *Model selection and estimation in the Gaussian graphical model.* Biometrika, Vol.94(1):19-35.

Examples

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

## run estimation
out1 = PreEst.auto(data, method="Banerjee06")
out2 = PreEst.auto(data, method="Lee17")

## Visualize
par(mfrow=c(1,3))
image(pracma::flipud(diag(10)),main="Original Covariance")
image(pracma::flipud(out1$C), main="Banerjee06")
image(pracma::flipud(out2$C), main="Lee17")
```

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 and Natsoulis G (2006). "Convex optimization techniques for fitting sparse Gaussian graphical models." In *Proceedings of the 23rd international conference on Machine learning*, pp. 89–96. ISBN 978-1-59593-383-6, doi: 10.1145/1143844.1143856, `http://portal.acm.org/citation.cfm?doid=1143844.1143856`.

Yuan M and Lin Y (2007). "Model Selection and Estimation in the Gaussian Graphical Model." *Biometrika*, **94**(1), pp. 19–35. ISSN 00063444, `http://www.jstor.org/stable/20441351`.

Friedman J, Hastie T and Tibshirani R (2008). "Sparse inverse covariance estimation with the graphical lasso." *Biostatistics*, **9**(3), pp. 432–441. ISSN 1465-4644, 1468-4357, doi: 10.1093/biostatistics/kxm045, `https://academic.oup.com/biostatistics/article-lookup/doi/10.1093/biostatistics/kxm045`.

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))
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