Package ‘bspcov’

July 16, 2023

Type Package

Title Bayesian Sparse Estimation of a Covariance Matrix

Version 1.0.0

Date 2023-07-16

Author Kwangmin Lee [aut],
Kyoungjae Lee [aut],
Seongil Jo [aut, cre],
Jaeyong Lee [ctb]

Maintainer Seongil Jo <bstatsjo@gamil.com>

Description Provides functions which perform Bayesian estimations of a covariance matrix for multivariate normal data. Assumes that the covariance matrix is sparse or band matrix and positive-definite. This software has been developed using funding supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (RS-2023-00211979, NRF-2022R1A5A7033499, NRF-2020R1A4A1018207 and NRF-2020R1C1C1A01013338).

Imports GIGrvg, coda, progress, BayesFactor, MASS, mvnfast, matrixcalc, matrixStats, purrr, dplyr, RSpectra, Matrix, plyr, CholWishart, magrittr, future, furrr, ks, ggplot2, coda, ggmcmc

License GPL-2

LazyLoad yes

URL https://github.com/statjs/bspcov

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

R topics documented:

`bandPPP` *Bayesian Estimation of a Banded Covariance Matrix*

Description

Provides a post-processed posterior for Bayesian inference of a banded covariance matrix.

Usage

```
bandPPP(X, k, eps, prior = list(), nsample = 2000)
```

Arguments

`X`	a n $\times$ p data matrix with column mean zero.
`k`	a scalar value (natural number) specifying the bandwidth of covariance matrix.
`eps`	a small positive number decreasing to 0 with default value $(log(k))^2 * (k + log(p))/n$.
`prior`	a list giving the prior information. The list includes the following parameters (with default values in parentheses): `A (I)` giving the positive definite scale matrix for the inverse-Wishart prior, `nu (p + k)` giving the degree of freedom of the inverse-Wishar prior.
`nsample`	a scalar value giving the number of the post-processed posterior samples.

Details

Lee, Lee, and Lee (2023+) proposed a two-step procedure generating samples from the post-processed posterior for Bayesian inference of a banded covariance matrix:

- Initial posterior computing step: Generate random samples from the following initial posterior obtained by using the inverse-Wishart prior $IW_p(B_0, \nu_0)$

$$\Sigma \mid X_1, \ldots, X_n \sim IW_p(B_0 + nS_n, \nu_0 + n),$$

  where $S_n = n^{-1} \sum_{i=1}^{n} X_i X_i^\top$.
- Post-processing step: Post-process the samples generated from the initial samples

$$\Sigma_{(i)} := \begin{cases} B_k(\Sigma^{(i)}) + \left[\epsilon_n - \lambda_{\min}\{B_k(\Sigma^{(i)})\}\right] I_p, & \text{if } \lambda_{\min}\{B_k(\Sigma^{(i)})\} < \epsilon_n, \\ B_k(\Sigma^{(i)}), & \text{otherwise ,} \end{cases}$$

where $\Sigma^{(1)}, \ldots, \Sigma^{(N)}$ are the initial posterior samples, ϵ_n is a small positive number decreasing to 0 as $n \to \infty$, and $B_k(B)$ denotes the k-band operation given as

$$B_k(B) = \{b_{ij} I(|i - j| \le k)\} \text{ for any } B = (b_{ij}) \in R^{p \times p}.$$

For more details, see Lee, Lee and Lee (2023+).

Value

Sigma
: a nsample $\times$ p(p+1)/2 matrix including lower triangular elements of covariance matrix.

p
: dimension of covariance matrix.

Author(s)

Kwangmin Lee

References

Lee, K., Lee, K., and Lee, J. (2023+), "Post-processes posteriors for banded covariances", *Bayesian Analysis*, DOI: 10.1214/22-BA1333.

See Also

cv.bandPPP estimate

Examples

```
## Not run:
n <- 25
p <- 50
Sigma0 <- diag(1, p)
X <- MASS::mvrnorm(n = n, mu = rep(0, p), Sigma = Sigma0)
res <- bspcov::bandPPP(X,2,0.01,nsample=100)
## End(Not run)
```

---

bmspcov *Bayesian Sparse Covariance Estimation*

Description

Provides a Bayesian sparse and positive definite estimate of a covariance matrix via the beta-mixture shrinkage prior.

Usage

```
bmspcov(X, Sigma, prior = list(), nsample = list())
```

Arguments

X
: a n $\times$ p data matrix with column mean zero.

Sigma
: an initial guess for Sigma.

prior
: a list giving the prior information. The list includes the following parameters (with default values in parentheses): `a (1/2)` and `b (1/2)` giving the shape parameters for beta distribution, `lambda (1)` giving the hyperparameter for the diagonal elements, `tau1sq (10000/(n*p^4))` giving the hyperparameter for the shrinkage prior of covariance.

`nsample` a list giving the MCMC parameters. The list includes the following integers (with default values in parentheses): `burnin (1000)` giving the number of MCMC samples in transition period, `nmc (1000)` giving the number of MCMC samples for analysis.

Details

Lee, Jo and Lee (2022) proposed the beta-mixture shrinkage prior for estimating a sparse and positive definite covariance matrix. The beta-mixture shrinkage prior for $\Sigma = (\sigma_{jk})$ is defined as

$$\pi(\Sigma) = \frac{\pi^u(\Sigma)I(\Sigma \in C_p)}{\pi^u(\Sigma \in C_p)},\ C_p = \{\ \text{all}\ p \times p\ \text{positive definite matrices}\ \},$$

where $\pi^u(\cdot)$ is the unconstrained prior given by

$$\pi^u(\sigma_{jk} \mid \rho_{jk}) = N\left(\sigma_{jk} \mid 0, \frac{\rho_{jk}}{1-\rho_{jk}}\tau_1^2\right)$$

$$\pi^u(\rho_{jk}) = Beta(\rho_{jk} \mid a, b),\ \rho_{jk} = 1 - 1/(1+\phi_{jk})$$

$$\pi^u(\sigma_{jj}) = Exp(\sigma_{jj} \mid \lambda).$$

For more details, see Lee, Jo and Lee (2022).

Value

`Sigma` a nmc $\times$ p(p+1)/2 matrix including lower triangular elements of covariance matrix.

`Phi` a nmc $\times$ p(p+1)/2 matrix including shrinkage parameters corresponding to lower triangular elements of covariance matrix.

`p` dimension of covariance matrix.

Author(s)

Kyoungjae Lee and Seongil Jo

References

Lee, K., Jo, S., and Lee, J. (2022), "The beta-mixture shrinkage prior for sparse covariances with near-minimax posterior convergence rate", *Journal of Multivariate Analysis*, 192, 105067.

See Also

sbmspcov

Examples

```
## Not run:
set.seed(1)
n <- 100
p <- 20

# generate a sparse covariance matrix:
True.Sigma <- matrix(0, nrow = p, ncol = p)
diag(True.Sigma) <- 1
Values <- -runif(n = p*(p-1)/2, min = 0.2, max = 0.8)
```

```
nonzeroIND <- which(rbinom(n=p*(p-1)/2,1,prob=1/p)==1)
zeroIND = (1:(p*(p-1)/2))[-nonzeroIND]
Values[zeroIND] <- 0
True.Sigma[lower.tri(True.Sigma)] <- Values
True.Sigma[upper.tri(True.Sigma)] <- t(True.Sigma)[upper.tri(True.Sigma)]
if(min(eigen(True.Sigma)$values) <= 0){
  delta <- -min(eigen(True.Sigma)$values) + 1.0e-5
  True.Sigma <- True.Sigma + delta*diag(p)
}

# generate a data
X <- MASS::mvrnorm(n = n, mu = rep(0, p), Sigma = True.Sigma)

# compute sparse, positive covariance estimator:
fout <- bspcov::bmspcov(X = X, Sigma = diag(diag(cov(X))))
post.est.m <- bspcov::estimate(fout)
sqrt(mean((post.est.m - True.Sigma)^2))
sqrt(mean((cov(X) - True.Sigma)^2))
## End(Not run)
```

`cv.bandPPP` *CV for Bayesian Estimation of a Banded Covariance Matrix*

Description

Performs leave-one-out cross-validation (LOOCV) to calculate the predictive log-likelihood for a post-processed posterior of a banded covariance matrix and selects the optimal parameters.

Usage

```
cv.bandPPP(X, kvec, epsvec, prior = list(), nsample = 2000, ncores = 2)
```

Arguments

`X`	a n × p data matrix with column mean zero.
`kvec`	a vector of natural numbers specifying the bandwidth of covariance matrix.
`epsvec`	a vector of small positive numbers decreasing to 0.
`prior`	a list giving the prior information. The list includes the following parameters (with default values in parentheses): `A (I)` giving the positive definite scale matrix for the inverse-Wishart prior, `nu (p + k)` giving the degree of freedom of the inverse-Wishar prior.
`nsample`	a scalar value giving the number of the post-processed posterior samples.
`ncores`	a scalar value giving the number of CPU cores.

Details

The predictive log-likelihood for each k and ϵ_n is estimated as follows:

$$\sum_{i=1}^{n} \log S^{-1} \sum_{s=1}^{S} p(X_i \mid B_k^{(\epsilon_n)}(Sigma_{i,s})),$$

where X_i is the ith observation, $\Sigma_{i,s}$ is the sth posterior sample based on $(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$, and $B_k^{(\epsilon_n)}$ represents the banding post-processing function. For more details, see (3) in Lee, Lee and Lee (2023+).

Value

`elpd`	a M $\times$ 3 dataframe having the expected log predictive density (ELPD) for each k and eps, where M = length(kvec) * length(epsvec).

Author(s)

Kwangmin Lee

References

Lee, K., Lee, K., and Lee, J. (2023+), "Post-processes posteriors for banded covariances", *Bayesian Analysis*, DOI: 10.1214/22-BA1333.

Gelman, A., Hwang, J., and Vehtari, A. (2014). "Understanding predictive information criteria for Bayesian models." *Statistics and computing*, 24(6), 997-1016.

See Also

bandPPP

Examples

```
## Not run:
Sigma0 <- diag(1,50)
X <- mvtnorm::rmvnorm(25,sigma = Sigma0)
kvec <- 1:2
epsvec <- c(0.01,0.05)
res <- bspcov::cv.bandPPP(X,kvec,epsvec,nsample=10,ncores=4)
plot(res)
## End(Not run)
```

`cv.thresPPP` *CV for Bayesian Estimation of a Sparse Covariance Matrix*

Description

Performs cross-validation to estimate spectral norm error for a post-processed posterior of a sparse covariance matrix.

Usage

```
cv.thresPPP(
  X,
  thresvec,
  epsvec,
  hyperparam = NULL,
  thresfun = "hard",
```

```
  nsample = 2000,
  ncores = 2
)
```

Arguments

`X`	a n $\times$ p data matrix with column mean zero
`thresvec`	a vector of real numbers specifying the parameter of the threshold function.
`epsvec`	a vector of small positive numbers decreasing to 0.
`thresfun`	a string to specify the type of threshold function. `fun ('hard')` giving the thresholding function ('hard' or 'soft') for the thresholding PPP procedure.
`nsample`	a scalar value giving the number of the post-processed posterior samples.
`ncores`	a scalar value giving the number of CPU cores.
`prior`	a list giving the prior information. The list includes the following parameters (with default values in parentheses): `A (I)` giving the positive definite scale matrix for the inverse-Wishart prior, `nu (p + 1)` giving the degree of freedom of the inverse-Wishar prior.

Details

Given a set of train data and validation data, the spectral norm error for each γ and ϵ_n is estimated as follows:

$$||\hat{\Sigma}(\gamma, \epsilon_n)^{(train)} - S^{(val)}||_2$$

where $\hat{\Sigma}(\gamma, \epsilon_n)_{train}$ is the estimate for the covariance based on the train data and $S^{(val)}$ is the sample covariance matrix based on the validation data. The spectral norm error is estimated by the 10-fold cross-validation. For more details, see the first paragraph on page 9 in Lee and Lee (2023).

Value

`CVdf`	a M $\times$ 3 dataframe having the estimated spectral norm error for each thres and eps, where M = length(thresvec) * length(epsvec)

Author(s)

Kwangmin Lee

References

Lee, K. and Lee, J. (2023), "Post-processes posteriors for sparse covariances", *Journal of Econometrics*, 236(3), 105475.

See Also

thresPPP

Examples

```
## Not run:
Sigma0 <- diag(1,50)
X <- mvtnorm::rmvnorm(25,sigma = Sigma0)
thresvec <- c(0.01,0.1)
epsvec <- c(0.01,0.1)
```

```
res <- bspcov::cv.thresPPP(X,thresvec,epsvec,nsample=100)
plot(res)
## End(Not run)
```

---

estimate *Point-estimate of posterior distribution*

Description

Compute the point estimate (mean) to describe posterior distribution.

Usage

```
estimate(object, ...)

## S3 method for class 'bspcov'
estimate(object, ...)
```

Arguments

object	an object from **bandPPP**, **bmspcov**, **sbmspcov**, and **thresPPP**.

Author(s)

Seongil Jo

See Also

plot.postmean.bspcov

Examples

```
## Not run:
n <- 25
p <- 50
Sigma0 <- diag(1, p)
X <- MASS::mvrnorm(n = n, mu = rep(0, p), Sigma = Sigma0)
res <- bspcov::bandPPP(X,2,0.01,nsample=100)
est <- bspcov::estimate(res)
## End(Not run)
```

plot.bspcov *Plot Diagnostics of Posterior Samples and Cross-Validation*

Description

Provides a trace plot of posterior samples and a plot of a learning curve for cross-validation

Usage

```
## S3 method for class 'bspcov'
plot(object, cols, rows, ...)
```

Arguments

object	an object from **bmspcov**, **sbmspcov**, **cv.bandPPP**, and **cv.thresPPP**.
cols	a scalar or a vector including specific column indices for the trace plot.
rows	a scalar or a vector including specific row indices greater than or equal to columns indices for the trace plot.

Author(s)

Seongil Jo

Examples

```
## Not run:
set.seed(1)
n <- 100
p <- 20

# generate a sparse covariance matrix:
True.Sigma <- matrix(0, nrow = p, ncol = p)
diag(True.Sigma) <- 1
Values <- -runif(n = p*(p-1)/2, min = 0.2, max = 0.8)
nonzeroIND <- which(rbinom(n=p*(p-1)/2,1,prob=1/p)==1)
zeroIND = (1:(p*(p-1)/2))[-nonzeroIND]
Values[zeroIND] <- 0
True.Sigma[lower.tri(True.Sigma)] <- Values
True.Sigma[upper.tri(True.Sigma)] <- t(True.Sigma)[upper.tri(True.Sigma)]
if(min(eigen(True.Sigma)$values) <= 0){
  delta <- -min(eigen(True.Sigma)$values) + 1.0e-5
  True.Sigma <- True.Sigma + delta*diag(p)
}

# generate a data
X <- MASS::mvrnorm(n = n, mu = rep(0, p), Sigma = True.Sigma)

# compute sparse, positive covariance estimator:
fout <- bspcov::sbmspcov(X = X, Sigma = diag(diag(cov(X))))
plot(fout, cols = c(1, 3, 4), rows = c(1, 3, 4))
#plot(fout, cols = 1, rows = 1:3)

# Cross-Validation for Banded Structure
```

```
Sigma0 <- diag(1,50)
X <- mvtnorm::rmvnorm(25,sigma = Sigma0)
kvec <- 1:2
epsvec <- c(0.01,0.05)
res <- bspcov::cv.bandPPP(X,kvec,epsvec,nsample=10,ncores=4)
plot(res)
## End(Not run)
```

`plot.postmean.bspcov` *Draw a Heat Map for Point Estimate of Covariance Matrix*

Description

Provides a heat map for posterior mean estimate of sparse covariance matrix

Usage

```
## S3 method for class 'postmean.bspcov'
plot(object, ...)
```

Arguments

`object` an object from **estimate**.

Author(s)

Seongil Jo

See Also

estimate

Examples

```
## Not run:
n <- 25
p <- 50
Sigma0 <- diag(1, p)
X <- MASS::mvrnorm(n = n, mu = rep(0, p), Sigma = Sigma0)
res <- bspcov::thresPPP(X, eps=0.01, thres=list(value=0.5,fun='hard'), nsample=100)
est <- bspcov::estimate(res)
plot(est)
## End(Not run)
```

`sbmspcov` *Bayesian Sparse Covariance Estimation using Sure Screening*

Description

Provides a Bayesian sparse and positive definite estimate of a covariance matrix via screened beta-mixture prior.

Usage

```
sbmspcov(X, Sigma, cutoff = list(), prior = list(), nsample = list())
```

Arguments

`X`	a n $\times$ p data matrix with column mean zero.
`Sigma`	an initial guess for Sigma.
`cutoff`	a list giving the information for the threshold. The list includes the following parameters (with default values in parentheses): `method ('FNR')` giving the method for determining the threshold value (method='FNR' uses the false negative rate (FNR)-based approach, method='corr' chooses the threshold value by sample correlations), `rho` a lower bound of meaningfully large correlations whose the defaults values are 0.25 and 0.2 for method = 'FNR' and method = 'corr', respectively. Note. If method='corr', `rho` is used as the threshold value. `FNR (0.05)` giving the prespecified FNR level when method = 'FNR'. `nsimdata (1000)` giving the number of simulated datasets for calculating Jeffreys' default Bayes factors when method = 'FNR'.
`prior`	a list giving the prior information. The list includes the following parameters (with default values in parentheses): `a (1/2)` and `b (1/2)` giving the shape parameters for beta distribution, `lambda (1)` giving the hyperparameter for the diagonal elements, `tau1sq (log(p)/(p^2*n))` giving the hyperparameter for the shrinkage prior of covariance.
`nsample`	a list giving the MCMC parameters. The list includes the following integers (with default values in parentheses): `burnin (1000)` giving the number of MCMC samples in transition period, `nmc (1000)` giving the number of MCMC samples for analysis.

Details

Lee, Jo, Lee, and Lee (2022+) proposed the screened beta-mixture shrinkage prior for estimating a sparse and positive definite covariance matrix. The screened beta-mixture shrinkage prior for $\Sigma = (\sigma_{jk})$ is defined as

$$\pi(\Sigma) = \frac{\pi^u(\Sigma) I(\Sigma \in C_p)}{\pi^u(\Sigma \in C_p)},\ C_p = \{\text{ all } p \times p \text{ positive definite matrices } \},$$

where $\pi^u(\cdot)$ is the unconstrained prior given by

$$\pi^u(\sigma_{jk} \mid \psi_{jk}) = N\left(\sigma_{jk} \mid 0, \frac{\psi_{jk}}{1-\psi_{jk}}\tau_1^2\right),\ \psi_{jk} = 1 - 1/(1+\phi_{jk})$$

$$\pi^u(\psi_{jk}) = Beta(\psi_{jk} \mid a, b) \text{ for } (j,k) \in S_r(\hat{R})$$

$$\pi^u(\sigma_{jj}) = Exp(\sigma_{jj} \mid \lambda),$$

where $S_r(\hat{R}) = \{(j,k) : 1 < j < k \leq p, |\hat{\rho}_{jk}| > r\}$, $\hat{\rho}_{jk}$ are sample correlations, and r is a threshold given by user.

For more details, see Lee, Jo, Lee and Lee (2022+).

Value

`Sigma`	a nmc $\times$ p(p+1)/2 matrix including lower triangular elements of covariance matrix.
`p`	dimension of covariance matrix.
`Phi`	a nmc $\times$ p(p+1)/2 matrix including shrinkage parameters corresponding to lower triangular elements of covariance matrix.
`INDzero`	a list including indices of off-diagonal elements screened by sure screening.
`cutoff`	the cutoff value specified by FNR-approach.

Author(s)

Kyoungjae Lee and Seongil Jo

References

Lee, K., Jo, S., Lee, K., and Lee, J. (2022+), "Scalable and optimal Bayesian inference for sparse covariance matrices via screened beta-mixture prior", arXiv:2206.12773.

See Also

bmspcov

Examples

```
## Not run:
set.seed(1)
n <- 100
p <- 20

# generate a sparse covariance matrix:
True.Sigma <- matrix(0, nrow = p, ncol = p)
diag(True.Sigma) <- 1
Values <- -runif(n = p*(p-1)/2, min = 0.2, max = 0.8)
nonzeroIND <- which(rbinom(n=p*(p-1)/2,1,prob=1/p)==1)
zeroIND = (1:(p*(p-1)/2))[-nonzeroIND]
Values[zeroIND] <- 0
True.Sigma[lower.tri(True.Sigma)] <- Values
True.Sigma[upper.tri(True.Sigma)] <- t(True.Sigma)[upper.tri(True.Sigma)]
if(min(eigen(True.Sigma)$values) <= 0){
  delta <- -min(eigen(True.Sigma)$values) + 1.0e-5
  True.Sigma <- True.Sigma + delta*diag(p)
}

# generate a data
X <- MASS::mvrnorm(n = n, mu = rep(0, p), Sigma = True.Sigma)

# compute sparse, positive covariance estimator:
```

```
fout <- bspcov::sbmspcov(X = X, Sigma = diag(diag(cov(X))))
post.est.m <- bspcov::estimate(fout)
sqrt(mean((post.est.m - True.Sigma)^2))
sqrt(mean((cov(X) - True.Sigma)^2))
## End(Not run)
```

`summary.bspcov` *Summary of Posterior Distribution*

Description

Provides the summary statistics for posterior samples of covariance matrix.

Usage

```
## S3 method for class 'bspcov'
summary(object, cols, rows, ...)
```

Arguments

`object`	an object from **bandPPP**, **bmspcov**, **sbmspcov**, and **thresPPP**.
`cols`	a scalar or a vector including specific column indices.
`rows`	a scalar or a vector including specific row indices greater than or equal to columns indices.

Note

If both `cols` and `rows` are vectors, they must have the same length.

Author(s)

Seongil Jo

Examples

```
## Not run:
set.seed(1)
n <- 100
p <- 20

# generate a sparse covariance matrix:
True.Sigma <- matrix(0, nrow = p, ncol = p)
diag(True.Sigma) <- 1
Values <- -runif(n = p*(p-1)/2, min = 0.2, max = 0.8)
nonzeroIND <- which(rbinom(n=p*(p-1)/2,1,prob=1/p)==1)
zeroIND = (1:(p*(p-1)/2))[-nonzeroIND]
Values[zeroIND] <- 0
True.Sigma[lower.tri(True.Sigma)] <- Values
True.Sigma[upper.tri(True.Sigma)] <- t(True.Sigma)[upper.tri(True.Sigma)]
if(min(eigen(True.Sigma)$values) <= 0){
  delta <- -min(eigen(True.Sigma)$values) + 1.0e-5
  True.Sigma <- True.Sigma + delta*diag(p)
```

```
}

# generate a data
X <- MASS::mvrnorm(n = n, mu = rep(0, p), Sigma = True.Sigma)

# compute sparse, positive covariance estimator:
fout <- bspcov::sbmspcov(X = X, Sigma = diag(diag(cov(X))))
summary(fout, cols = c(1, 3, 4), rows = c(1, 3, 4))
#summary(fout, cols = 1, rows = 1:p)
## End(Not run)
```

`thresPPP` *Bayesian Estimation of a Sparse Covariance Matrix*

Description

Provides a post-processed posterior (PPP) for Bayesian inference of a sparse covariance matrix.

Usage

```
thresPPP(X, eps, thres = list(), prior = list(), nsample = 2000)
```

Arguments

`X`	a n $\times$ p data matrix with column mean zero.
`eps`	a small positive number decreasing to 0.
`thres`	a list giving the information for thresholding PPP procedure. The list includes the following parameters (with default values in parentheses): `value (0.1)` giving the positive real number for the thresholding PPP procedure, `fun ('hard')` giving the thresholding function ('hard' or 'soft') for the thresholding PPP procedure.
`prior`	a list giving the prior information. The list includes the following parameters (with default values in parentheses): `A (I)` giving the positive definite scale matrix for the inverse-Wishart prior, `nu (p + 1)` giving the degree of freedom of the inverse-Wishar prior.
`nsample`	a scalar value giving the number of the post-processed posterior samples.

Details

Lee and Lee (2023) proposed a two-step procedure generating samples from the post-processed posterior for Bayesian inference of a sparse covariance matrix:

- Initial posterior computing step: Generate random samples from the following initial posterior obtained by using the inverse-Wishart prior $IW_p(B_0, \nu_0)$

$$\Sigma \mid X_1, \ldots, X_n \sim IW_p(B_0 + nS_n, \nu_0 + n),$$

  where $S_n = n^{-1} \sum_{i=1}^n X_i X_i^\top$.
- Post-processing step: Post-process the samples generated from the initial samples

$$\Sigma_{(i)} := \begin{cases} H_{\gamma_n}(\Sigma^{(i)}) + \left[\epsilon_n - \lambda_{\min}\{H_{\gamma_n}(\Sigma^{(i)})\}\right] I_p, & \text{if } \lambda_{\min}\{H_{\gamma_n}(\Sigma^{(i)})\} < \epsilon_n, \\ H_{\gamma_n}(\Sigma^{(i)}), & \text{otherwise }, \end{cases}$$

where $\Sigma^{(1)}, \ldots, \Sigma^{(N)}$ are the initial posterior samples, ϵ_n is a positive real number, and $H_{\gamma_n}(\Sigma)$ denotes the generalized threshodling operator given as

$$(H_{\gamma_n}(\Sigma))_{ij} = \begin{cases} \sigma_{ij}, & \text{if } i = j, \\ h_{\gamma_n}(\sigma_{ij}), & \text{if } i \neq j, \end{cases}$$

where σ_{ij} is the (i,j) element of Σ and $h_{\gamma_n}(\cdot)$ is a generalized thresholding function.

For more details, see Lee and Lee (2023).

Value

`Sigma` a nsample $\times$ p(p+1)/2 matrix including lower triangular elements of covariance matrix.

`p` dimension of covariance matrix.

Author(s)

Kwangmin Lee

References

Lee, K. and Lee, J. (2023), "Post-processes posteriors for sparse covariances", *Journal of Econometrics*.

See Also

cv.thresPPP

Examples

```
## Not run:
n <- 25
p <- 50
Sigma0 <- diag(1, p)
X <- MASS::mvrnorm(n = n, mu = rep(0, p), Sigma = Sigma0)
res <- bspcov::thresPPP(X, eps=0.01, thres=list(value=0.5,fun='hard'), nsample=100)
est <- bspcov::estimate(res)
## End(Not run)
```

Index