Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

bspcov: An R Package for Bayesian sparse covariance matrix estimation

Kyeongwon Lee [a], Kyoungjae Lee [b], Seongil Jo [c,*], Kwangmin Lee [d,*]

[a] *Department of Mathematics, University of Maryland, United States of America*
[b] *Department of Statistics, Sungkyunkwan University, Republic of Korea*
[c] *Department of Statistics, Inha University, Republic of Korea*
[d] *Department of Bigdata Convergence, Chonnam National University, Republic of Korea*

ARTICLE INFO

Keywords:
Beta-mixture shrinkage
Post-processing
Posterior inference

ABSTRACT

The bspcov R package provides a Bayesian inference for covariance matrices. The bspcov is developed to aid in research that involves estimating constrained covariance matrices by enabling the use of state-of-the-art Bayesian inference methods. It consists of the main functions bmspcov, sbmspcov, bandPPP and thresPPP that conduct posterior inference for sparse or banded covariance matrices. The functions bmspcov and sbmspcov implement block Gibbs samplers based on beta-mixture and screened beta-mixture shrinkage priors, respectively. The functions bandPPP and thresPPP implement a direct posterior sampling from the post-processed posterior for banded and sparse covariance matrices. We demonstrate how to use the main functions with real data applications.

Code metadata

Current code version	v1.0.3
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-25-00450
Permanent link to Reproducible Capsule	Reproducible examples provided as part of the documentation and https://github.com/statjs/bspcov/tree/v1.0.3/examples
Legal Code License	GNU General Public License (GPL) v2
Code versioning system used	Git
Software code languages, tools, and services used	R
Compilation requirements, operating environments & dependencies	R $\geq$ 4.2.0
If available Link to developer documentation/manual	https://github.com/statjs/bspcov/releases/latest/download/bspcov-manual.pdf
Support email for questions	kwlee1718@gmail.com

The bspcov package is an open-source project, and its source code is publicly available at https://github.com/statjs/bspcov. The repository includes a README.txt, LICENSE.txt, source code in the src/ directory, and documentation in the docs/ directory. You can also download specific versions of the package as tagged releases, such as bspcov_x.y.z.tar.gz, from the releases page.

Additionally, the bspcov package can be installed from CRAN:

```r
install.packages("bspcov")
```

or from GitHub (statjs/bspcov), by using the function install_github() from the devtools package:

```r
devtools::install_github("statjs/bspcov",
    ref = "main")
```

1. Motivation and significance

Estimating the covariance matrix is a crucial but challenging task for discovering the dependence structure between variables or conducting the best linear prediction based on multivariate data. In high-dimensional settings, where the number of variables p is in the hundreds or thousands, the number of parameters in a covariance matrix grows quadratically. To address this, structural assumptions such as sparsity or bandedness are commonly imposed, meaning most off-diagonal entries are assumed to be zero [1–5].

Various R packages have been developed for the estimation of high-dimensional covariance or precision matrices, including the glasso

* Corresponding authors.
E-mail addresses: bstatsjo@gmail.com (Seongil Jo), klee564@jnu.ac.kr (Kwangmin Lee).

https://doi.org/10.1016/j.softx.2025.102338
Received 30 June 2025; Received in revised form 16 August 2025; Accepted 27 August 2025
Available online 17 September 2025

package [6], `huge` package [7], `flare` package [8], `CovTools` package [9], `rags2ridges` package [10], and `spcov` package [11]. These packages primarily rely on penalized likelihood approaches to produce point estimators. However, frequentist methods often face limitations when it comes to quantifying uncertainty, which is critical in scientific decision-making.

Bayesian methods provide a natural framework for uncertainty quantification through the posterior distribution. Despite active research in Bayesian covariance estimation, software implementation has been limited. To the best of our knowledge, among publicly available R packages, only `BDgraph` [12] and `BGGM` [13] support Bayesian inference for precision matrices, while no existing package offers scalable Bayesian inference for high-dimensional covariance matrices.

In this paper, we introduce the `bspcov` R package, which implements recently proposed Bayesian methods for estimating sparse or banded covariance matrices. It supports continuous shrinkage prior models (beta-mixture [5] and screened beta-mixture [14]) as well as post-processed posterior (PPP) methods [4,15]. The package enables posterior sampling and inference for high-dimensional covariance matrices, providing both point estimates and uncertainty quantification.

The beta-mixture (BM) prior, proposed by [5], places beta-mixture and gamma priors on off-diagonal and diagonal entries, respectively. A block Gibbs sampler is developed based on a transformation inspired by [16], and the resulting posterior achieves a nearly minimax convergence rate. However, the computational complexity for sampling each column is $O(p^3)$, which can be substantial in high-dimensional settings.

To address this, the screened beta-mixture (SBM) prior was proposed by [14], which pre-screens weakly correlated variable pairs using sample correlations and excludes them from inference. This significantly reduces the computational burden while retaining strong signals, thereby enabling scalable Bayesian inference for sparse covariance matrices. Specifically, the computational complexity for one iteration of the SBM prior is $O(nps)$ per column, where s denotes the number of nonzero off-diagonal entries in a column.

The post-processed posterior (PPP) approaches [4,15] provide an alternative, sampling from a conjugate inverse-Wishart posterior in the first step (without constraints), and applying a banding or thresholding transformation in the second step to enforce structure. For the PPP methods, each posterior sample requires $O(p^3)$ due to the use of Bartlett decomposition, matrix multiplication, and matrix inversion. The PPP methods do not need the MCMC convergence diagnosis since the posterior sames are independent, which makes the PPP methods computationally efficient. Moreover, these methods are theoretically justified by achieving nearly minimax-optimal convergence rates under high-dimensional regimes where $\log p = o(n)$.

The `bspcov` package implements these Bayesian methods in a unified and user-friendly framework. It provides scalable posterior sampling routines for the BM and SBM priors and the PPP approaches. By combining theoretical guarantees with computational efficiency, `bspcov` offers a flexible platform for Bayesian inference of high-dimensional covariance matrices, allowing users to obtain both point estimates and uncertainty quantification under various structural assumptions.

2. Software description

The `bspcov` R package provides a Bayesian inference for covariance matrices, developed to aid in research that involves estimating constrained covariance matrices by enabling the use of state-of-the-art Bayesian inference methods.

2.1. Software architecture

The package contains four main functions for estimating sparse or banded covariance matrices.

- `bmspcov()`: This function provides a Bayesian sparse and positive definite estimate of a covariance matrix via the BM prior.
- `sbmspcov()`: This function provides a Bayesian sparse and positive definite estimate of a covariance matrix via the SBM prior.
- `bandPPP()`: This function provides a PPP for Bayesian inference of a banded covariance matrix.
- `thresPPP()`: This function provides a PPP for Bayesian inference of a sparse covariance matrix.

The software architecture of `bspcov` is illustrated in Fig. 1, while its specific functionalities and associated methods are detailed in Section 2.2.

2.2. Software functionalities

2.2.1. Main functions

The functions in `bspcov` package perform Bayesian estimations of a covariance matrix for multivariate normal data. Assumes that the covariance matrix is a sparse or banded matrix and positive-definite. The covariance estimation in function `bmspcov()` and `sbmspcov()` is based on the block Gibbs samplers with support for multiple MCMC chains and parallel processing. The `bandPPP()` and `thresPPP()` functions directly extract posterior samples from the post-processed posterior.

The function `bmspcov()` conducts the block Gibbs sampler based on the BM prior [5], which consists of independent BM for off-diagonal and gamma priors and diagonal entries. The detail for the BM prior is given in Appendix A. The syntax is

```
bmspcov(X, Sigma, prior, nsample, nchain, seed, do.parallel)
```

with seven arguments: `X`, `Sigma`, `prior`, `nsample`, `nchain`, `seed`, and `do.parallel`. Given these arguments, the function estimates the covariance matrix using the `Sigma` as the initial value for the MCMC algorithm, upon observing the centered $n \times p$ data matrix `X`. Here, the argument `prior` specifies the parameters of the prior distribution. The argument `prior` is a list of hyperparameters, including `a` and `b`, which specify the shape parameters of the beta distribution; `lambda`, which is the hyperparameter for the diagonal elements; and `tau1sq`, which is the hyperparameter for the shrinkage prior of the covariance. The default values of each hyperparameter, along with additional explanations, are provided in Appendix A. The argument `nsample` sets the number of samples including burn-in size `burnin` and the number of MCMC samples `nmc`. The argument `nchain` specifies the number of MCMC chains to run (default is 1), while `seed` sets the random seed for reproducibility (with each chain receiving `seed + i - 1` for chain i). The logical argument `do.parallel` enables parallel processing of multiple chains using `future.apply` when set to `TRUE` (default is `FALSE`). The function returns MCMC samples for the lower triangular elements of the covariance matrix and its shrinkage parameters from all chains, along with the dimension p and chain information.

The function `sbmspcov()` estimates the covariance matrix via the SBM prior, which is a continuous shrinkage prior combined with a screening process, as described by Lee et al. [14]. The syntax is

```
sbmspcov(X, Sigma, cutoff, prior, ...)
```

with eight arguments: the arguments of the `bmspcov()` function and `cutoff`. The argument `cutoff` provides information about the threshold, allowing users to choose a method for determining the threshold based on either the false negative rate (FNR) ((`cutoff$method = 'FNR'`) or on the correlation coefficient (`cutoff$method = 'corr'`)). The default setting for the FNR-based method is `list(rho = 0.25, FNR = 0.05, nsimdata = 1000)`. This corresponds to choosing the threshold such that, when Bayesian tests for zero correlation [17] are repeatedly conducted 1000 times using random samples of size n from a bivariate normal distribution with true correlation 0.25, the resulting false negative rate (FNR) is approximately 0.05. The

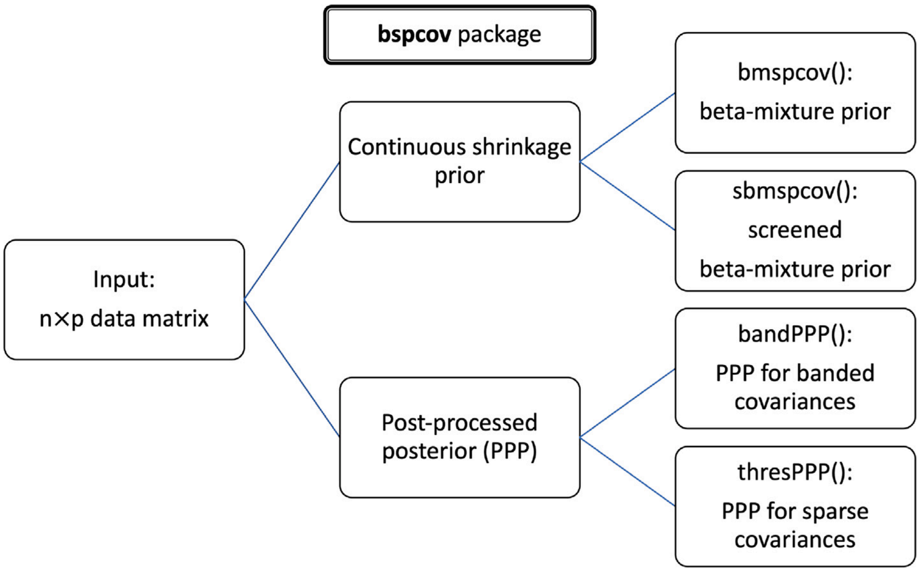


Fig. 1. A diagram of the bspcov package.

default setting for the correlation-based method is `list(thr = 0.2)`, which corresponds to selecting the 0.2-quantile of the absolute sample correlation coefficients as the threshold. This means that all off-diagonal entries with absolute sample correlations in the bottom 80% are set to zero, effectively retaining only the top 20% with the strongest correlations. Similar to `bmspcov()`, the `nchain`, `seed`, and `do.parallel` arguments control the number of chains, reproducibility, and parallel execution, respectively. The function returns similar results to the function `bmspcov()`, along with a list of off-diagonal elements screened by sure screening and cutoff value.

The `bandPPP()` and `thresPPP()` functions implement the PPP method to extract samples of the banded covariance matrix [4] and the sparse covariance matrix [15], respectively. The syntaxes are

$$\mathtt{bandPPP(X, k, eps, prior, nsample)}$$

and $\mathtt{thresPPP(X, eps, thres, prior, nsample)}$.

These functions take a centered $n \times p$ data matrix `X`, the parameters of the prior distribution `prior`, and the number of samples `nsample`. The (A, ν) in the initial prior distribution is specified by setting a list in the argument of `prior`. For example, if a user intends to set $A = I_p$ and $\nu = p + 1$, where I_p is the $p \times p$ identity matrix, then `prior = list(A=diag(1,p),nu=p+1)` can be used. For `bandPPP`, `k` and `eps` are the tuning parameters for the banding operator. For `thresPPP`, `thres` and `eps` are the tuning parameters for the generalized thresholding operator. These values are, by default, tuning parameters that must be specified by the user. However, since we provide functions (`cv.bandPPP` and `cv.thresPPP`) that select them using cross-validation methods, in most cases users do not need to specify them manually. Detailed descriptions of each operator are provided in Appendix A.

2.2.2. *Auxiliary functions*

We provide functions `cv.bandPPP` and `cv.thresPPP` to choose the tuning parameters in the banding and generalized thresholding operators, respectively, via cross-validation methods. The syntaxes are

$$\mathtt{cv.bandPPP(X, kvec, epsvec, prior, nsample, ncores)}$$

and $\mathtt{cv.thresPPP(X, thresvec, epsvec, prior, nsample, ncores)}$.

`cv.bandPPP` (`cv.thresPPP`) finds the best k (γ) and ϵ_n among the vectors `kvec` (`thresvec`) and `epsvec`. For example, if a user intends to consider $k = 1, 2, 3$ and $\epsilon_n = 0.01, 0.05$, then the following code works as intended:

```
cv.bandPPP(...,kvec=1:3,epsvec=c(0.01,0.05),...).
```

The cross-validation procedure can be conducted in parallel using the number of cores specified by the `ncores` argument.

We provide three functions for summarizing the posterior distribution. The function `estimate()` computes the point estimate (mean) of the posterior distribution. The function `quantile.bspcov()` computes quantiles of the posterior distribution at specified probability levels, which is useful for constructing element-wise credible intervals and assessing uncertainty. The function `summary.bspcov()` provides comprehensive summary statistics of the posterior samples. The `quantile.bspcov()` function computes quantiles for each element of the covariance matrix at user-specified probability levels. For multiple chains, it automatically combines samples from all chains to compute more robust quantiles. The default probability levels are `c(0.025, 0.5, 0.975)`, providing the element-wise median and 95% credible interval bounds. The function returns a list of quantile matrices, one for each specified probability level. The `summary.bspcov()` function returns a table of summary statistics including empirical mean, standard deviation, and customizable quantiles for specific indices (`rows` and `cols`) of the posterior samples. When multiple chains are used, the function additionally provides MCMC convergence diagnostics including effective sample size (`n_eff`) and potential scale reduction factor (`R-hat`) [18]. The syntaxes are

```
estimate(object),
quantile(object, probs),
```

and `summary(object, cols, rows, quantiles)`.

Additionally, we provide the functions `proc_colon()` and `proc_SP500()` to preprocess the colon and S&P 500 datasets, respectively. For the colon data, we select the 50 most significant genes (i.e. $p = 50$) based on the two-sample t-statistic. The syntax is:

```
proc_colon(colon, tissues)
```

Details and an example of using this function are presented in Section 3.1. For the S&P 500 data, we select stocks from specified sectors, compute monthly returns, and extract idiosyncratic components via factor adjustment. The syntax is:

```
proc_SP500(data, sectors)
```

An example of using this function is presented in Section 3.3.

Finally, we provide functions for visualization. The function `plot.bspcov()` plots trace plots of posterior samples and learning curves for cross-validation. For multiple chains, the trace plots display each chain in a different color to facilitate convergence assessment. The syntax is

`plot(object, cols, rows)`,

where `object` is an object from `bmspcov()`, `sbmspcov()`, `bandPPP()`, and `thresPPP()` functions, `cols` and `rows` are scalars or vectors specifying the indices of the covariance matrix elements to plot.

The function `plot.postmean.bspcov()` plots a heat map for the posterior mean estimate of the sparse covariance matrix with enhanced customization options. The function now provides flexible visualization with customizable color schemes, axis labels, and optional value display on tiles for smaller matrices. The syntax is

`plot(estimate, title, ...)`,

where `estimate` is an object from `estimate()`, `title` specifies the plot title, `color_low` and `color_high` define the color gradient (default is black to white), `x_label` and `y_label` specify the axis labels (default is "`Variable`"), and `show_values` is a logical flag for displaying numerical values on the heatmap tiles.

The function `plot.quantile.bspcov()` provides visualization for posterior quantiles computed by `quantile()`. It produces heatmap visualizations showing the covariance structure at different quantile levels, which is particularly useful for assessing uncertainty in the posterior distribution. When multiple quantiles are provided, the function arranges them in a grid layout using `patchwork` for easy comparison. The syntax is

`plot(quantile_object, type, ...)`,

where `quantile_object` is the output from `quantile()`, `type` specifies the visualization type (default is "`heatmap`"), `titles` provides custom titles for each quantile plot, `ncol` controls the number of columns in the layout, and `x_label` and `y_label` specify the axis labels (default is "`Variable`"). Additionally, the helper function `save_quantile_plot()` facilitates saving these visualizations with appropriate dimensions.

Based on these functions, various statistical inferences can be conducted on the sparse or banded covariance matrices, with comprehensive visualization support for both convergence diagnostics and uncertainty quantification. All visualization functions share consistent aesthetics and customization options, enabling coherent visual analysis across different aspects of the posterior distribution.

3. Illustrative examples

3.1. Sparse covariance estimation for gene expression data

In this subsection, we demonstrate sparse covariance matrix estimation using `bmspcov` and `sbmspcov` functions through linear discriminant analysis (LDA) [5,14]. The LDA classifier assigns each observation to one of two groups based on the following decision rule:

$$\delta_j(X) = \underset{j\in\{1,2\}}{\operatorname{argmax}} \left\{ X^\top \hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_j - \frac{1}{2}\hat{\boldsymbol{\mu}}_j^\top \hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_j + \log\hat{\omega}_j \right\},$$

where $\hat{\omega}_j$ and $\hat{\boldsymbol{\mu}}_j$ are the proportion and sample mean, respectively, for class $j \in \{1,2\}$. In LDA, estimating covariance is essential as seen in the formula above.

We use the colon cancer dataset, which includes gene expression values from 22 colon tumor tissues and 40 non-tumor tissues, and is available at http://genomics-pubs.princeton.edu/oncology/affydata/. As a preprocessing step before analyzing the data, we select the 50 most significant genes (denoted as $p = 50$) based on the two-sample t-statistic [5,14,19].

Table 1
Classification performance using 5-fold cross-validation.

	BM prior	SBM prior
Misclassification rate	0.113	0.162

Fig. 2 shows the posterior mean covariance estimates obtained by the beta-mixture shrinkage methods. The two methods exhibit different sparsity patterns: `bmspcov` employs continuous shrinkage across all elements, while `sbmspcov` combines screening with shrinkage, resulting in more aggressive sparsification of weak correlations.

Table 1 presents the classification performance evaluated through 5-fold cross-validation. The `bmspcov` method achieves lower misclassification rate, while `sbmspcov` shows slightly higher error rates due to its aggressive screening procedure. For comprehensive analysis including additional methods (`thresPPP`), experimental settings (burn-in period, MCMC sample sizes, chain configurations), MCMC convergence diagnostics, uncertainty quantification, and performance comparisons with other Bayesian packages (JAGS, Stan), see Appendix C.

We visualize the element-wise posterior uncertainty by computing and plotting the 2.5%, 50% (median), and 97.5% quantiles using the `quantile.bspcov()` and `plot.quantile.bspcov()` functions. Fig. 3 presents these uncertainty visualizations.

The `bmspcov` and `sbmspcov` methods produce similar sparse patterns with smooth transitions.

The following code demonstrates the usage of two methods:

```
# attach library
library(bspcov)

# fix random seed
set.seed(1234)

# load and preprocess data
data("colon")
data("tissues")
colon_data <- proc_colon(colon, tissues =
    tissues)
X <- colon_data$X

# beta-mixture prior (parallel processing
    automatically set up when do.parallel =
    TRUE)
foo.bmspcov <- bmspcov(X,
  Sigma = cov(X), nchain = 4,
  nsample = list(burnin = 1000, nmc = 2000),
  seed = 1234, do.parallel = TRUE
)

# screened beta-mixture prior (parallel
    processing automatically set up when do.
    parallel = TRUE)
cutoff <- list(method = "corr", rho = 0.25)
foo.sbmspcov <- sbmspcov(
  X = X, Sigma = cov(X), cutoff, nchain = 4,
  nsample = list(burnin = 1000, nmc = 2000),
  seed = 1234, do.parallel = TRUE
)

# posterior mean estimate
sigmah.bmspcov <- estimate(foo.bmspcov)
sigmah.sbmspcov <- estimate(foo.sbmspcov)
```

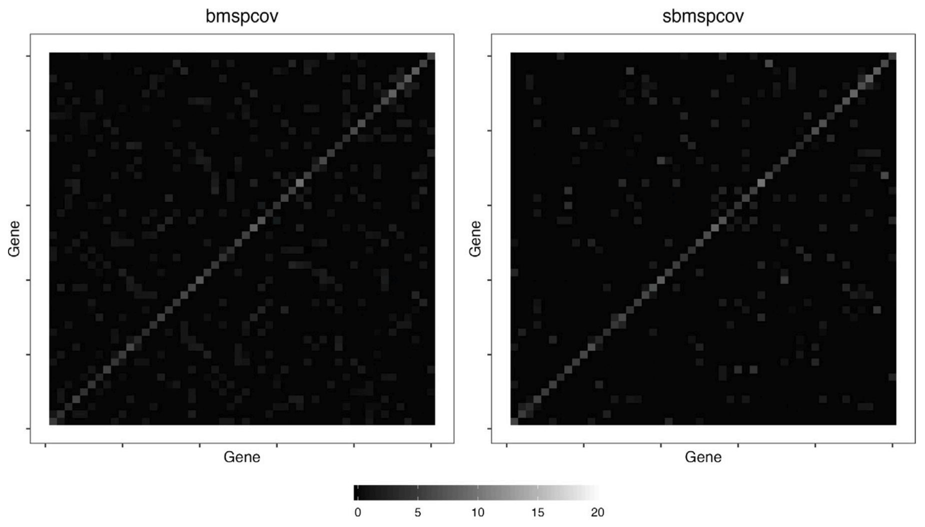
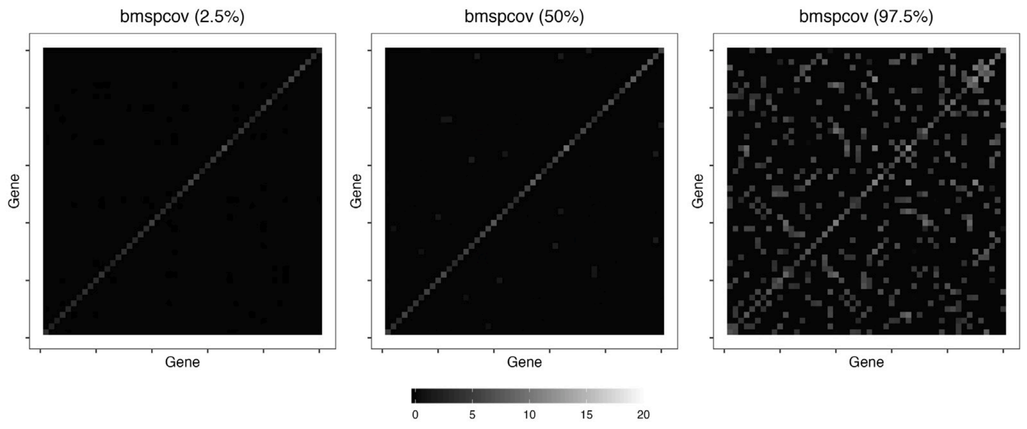
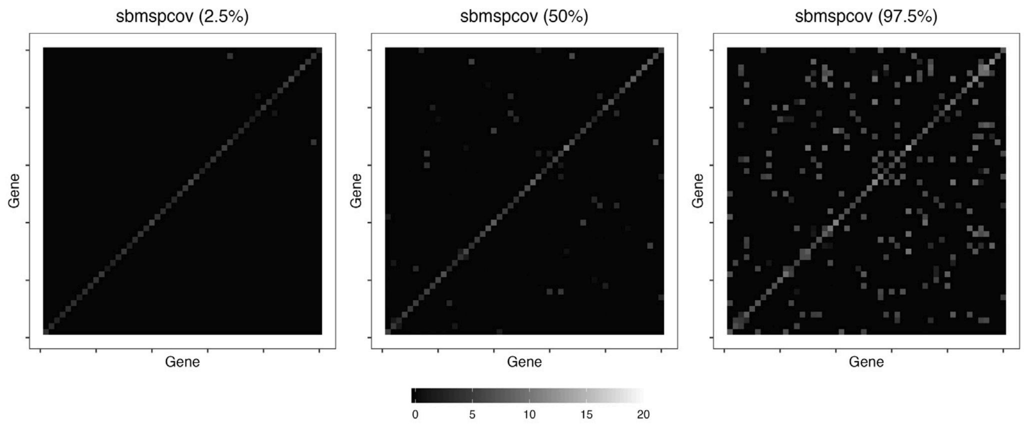


Fig. 2. Posterior mean covariance estimates for the colon cancer dataset: BM prior (`bmspcov`, left) and SBM prior (`sbmspcov`, right).



(a) `bmspcov`: Element-wise 2.5%, 50%, and 97.5% posterior quantiles



(b) `sbmspcov`: Element-wise 2.5%, 50%, and 97.5% posterior quantiles

Fig. 3. Uncertainty quantification across two methods: Comparison of element-wise posterior quantiles.

3.2. *Post-processed posterior for banded covariances*

We illustrate an application of the banding PPP to time series classification. Let $\boldsymbol{X} \in \mathbb{R}^p$ denote a time series for p time points. The $\boldsymbol{X}$ is assumed to be generated from either $N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ or $N(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, where $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are assumed to be banded covariances based on the time series assumption. Using the quadratic discriminant analysis (QDA), we classify an observation $\boldsymbol{X}$ as

$$\delta_j(\boldsymbol{X}) = \underset{j \in \{1,2\}}{\operatorname{argmax}} \Big\{ l(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j; \boldsymbol{X}) + \log(\pi_j) \Big\},$$

where $l(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j; \boldsymbol{X})$ is the log-likelihood function of the multivariate normal distribution with mean $\boldsymbol{\mu}_j$ and covariance $\boldsymbol{\Sigma}_j$, and π_j is the prior probability for the jth class. The QDA requires estimating $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, which can be estimated by the banding PPP method.

We apply the QDA to eRing data from [20]. The eRing data consists of electric field sensing data of six types of hand gestures. Each observation is a time series data for 65 time points, and 50 observations are provided for each gesture. In this illustration, we focus on the classification of two gestures.

Fig. 4 presents the posterior mean estimates of the banded covariance matrices for the two gestures, computed using the `estimate()` function. The figure clearly shows that both covariances exhibit banded structures, consistent with the temporal correlation patterns expected in time series data. To assess the uncertainty in these estimates, Fig. 5 visualizes the element-wise posterior uncertainty through the 2.5%, 50% (median), and 97.5% quantiles computed using the `quantile()` function. The quantile plots reveal higher uncertainty in the off-diagonal elements farther from the main diagonal.

We also have evaluated the classification performance using the 5-fold cross-validation method. The test errors are represented in Table 2. In this example, we set $\pi_1 = \pi_2 = 1/2$ and set $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ as the sample mean vectors.

The following are example codes for `bandPPP` function.

```r
# attach library
library(bspcov)
library(dplyr)

# fix random seed
set.seed(1234)

# load and preprocess data
library(mlmts)
data("ERing")
Ering_class1 <- ERing$data[which(ERing$
    classes == 1)] %>%
  purrr::map(~ as.vector(t(.x[, 1]))) %>%
  do.call("rbind", .)
Ering_class2 <- ERing$data[which(ERing$
    classes == 2)] %>%
  purrr::map(~ as.vector(t(.x[, 1]))) %>%
  do.call("rbind", .)


# estimation by band PPP
## gesture 1
cov1 <- bandPPP(scale(Ering_class1, scale =
    FALSE, center = TRUE), k = 3, eps =
    0.05)
postmean1 <- estimate(cov1)
quantile1 <- quantile(cov1, probs = c(0.025,
     0.5, 0.975))

## gesture 2
cov2 <- bandPPP(scale(Ering_class2, scale =
    FALSE, center = TRUE), k = 3, eps =
    0.05)
postmean2 <- estimate(cov2)
quantile2 <- quantile(cov2, probs = c(0.025,
     0.5, 0.975))
```

3.3. Post-processed posterior for sparse covariances

We illustrate an application of the threshold PPP to estimate the idiosyncratic covariance of S&P 500 data. We collect S&P 500 data from [21] with the collection period from Jan 2013 to Nov 2023. We excluded stocks with missing data and considered the stocks of the following sectors: Energy, financials, materials, real estate, utilities, and information technology. Let $\boldsymbol{Y}_i \in \mathbb{R}^p$ denote the excess return vectors of the ith month. Then, we have $n = 132$ observations for $p = 223$ stocks, which are denoted by $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n \in \mathbb{R}^p$.

Table 2
Contingency table of the classification for the gesture of the eRing data. The classification error is evaluated by 5-fold cross-validation method.

True class	Estimated class		
	Gesture 1	Gesture 2	Total
Gesture 1	1.00	0.00	1
Gesture 2	0.74	0.26	1

Suppose $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$ are generated from the approximate factor model [22] as

$$\boldsymbol{Y}_i = \boldsymbol{L}\boldsymbol{f}_i + \boldsymbol{u}_i,\ \boldsymbol{u}_i \sim N_p(\boldsymbol{0}_p, \boldsymbol{\Sigma}_u),\ i = 1, \ldots, n,$$

where $\boldsymbol{L} \in \mathbb{R}^{p\times K}$ is the factor loading matrix, K is the number of factors, and $\boldsymbol{\Sigma}_u \in \mathcal{C}_p$ is the idiosyncratic covariance. The $\boldsymbol{f}_i$ is the factor score of the ith observation and considered to be a random variable satisfying $\mathrm{E}(\boldsymbol{f}_i) = 0$ and $\mathrm{Cov}(\boldsymbol{f}_i) = \boldsymbol{I}_K$ for the identifaiblity. The factor loading matrix explains the volatilities induced by macroeconomic factors, and the idiosyncratic covariance explains firm-specific volatilities.

We estimate $\boldsymbol{\Sigma}_u$ using the threshold PPP method under the sparse assumption of $\boldsymbol{\Sigma}_u$. Since $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ are not observable, we substitute them with the estimates $\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_n$, which is obtained as the residual of the factor model. First, define $\hat{\boldsymbol{L}} \in \mathbb{R}^{p\times K}$ and $\hat{\boldsymbol{F}} = [\hat{\boldsymbol{f}}_1, \ldots, \hat{\boldsymbol{f}}_n]^T \in \mathbb{R}^{n\times K}$ as the constrained least square estimator,

$$(\hat{\boldsymbol{L}}, \hat{\boldsymbol{F}}) := \mathrm{argmin}_{\boldsymbol{L},\boldsymbol{F}} \|\mathbb{Y}_n^T - \boldsymbol{L}\boldsymbol{F}^T\|_F^2,\ n^{-1}\boldsymbol{F}^T\boldsymbol{F} = \boldsymbol{I}_K,\ \boldsymbol{L}^T\boldsymbol{L} \text{ is diagonal},$$

where $\mathbb{Y}_n^T = [\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n] \in \mathbb{R}^{p\times n}$. Then, the residual $\hat{\boldsymbol{u}}_i$ is defined as

$$\hat{\boldsymbol{u}}_i = \boldsymbol{Y}_i - \hat{\boldsymbol{L}}\hat{\boldsymbol{f}}_i^{\hat{K}},\ i = 1, \ldots, n.$$

We apply the thresholding post-processed posterior considering $\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_n$ as observations. The theoretical justification of this approach is given in Section 3 of [15]. We illustrate the posterior mean for $\boldsymbol{\Sigma}_u$ of the S&P 500 data in Fig. 6. The figure shows that the firm-specific volatilities are correlated for the stocks in the same sector.

The following are example codes for the estimation of the idiosyncratic covariance.

```r
library(bspcov)
library(dplyr)

# fix random seed
set.seed(1234)

# load and preprocess SP500 data
data("SP500")
sectors <- c("Energy", "Financials", "
    Materials", "Real Estate", "Utilities",
    "Information Technology")
SP500_idioerr <- proc_SP500(SP500, sectors)
Uhat <- SP500_idioerr$Uhat
sectornames <- SP500_idioerr$sectornames
PPPres <- thresPPP(Uhat, eps = 0, thres =
    list(value = 0.0020, fun = "hard"),
    nsample = 100)

# apply thresPPP
postmean <- estimate(PPPres)
```

4. Impact

The `bspcov` package enables efficient Bayesian estimation of high-dimensional covariance matrices with structural assumptions such as sparsity or bandedness. This allows researchers to explore new models and apply Bayesian inference in areas where existing methods were

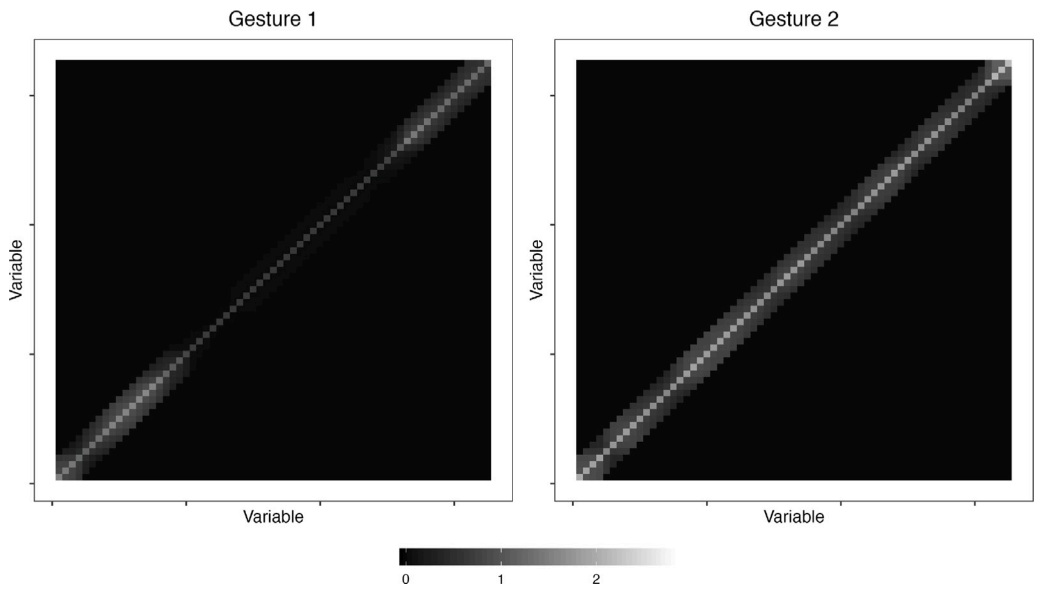


Fig. 4. The covariances of sensor data for two gestures. The covariances are estimated by the banding PPP method.

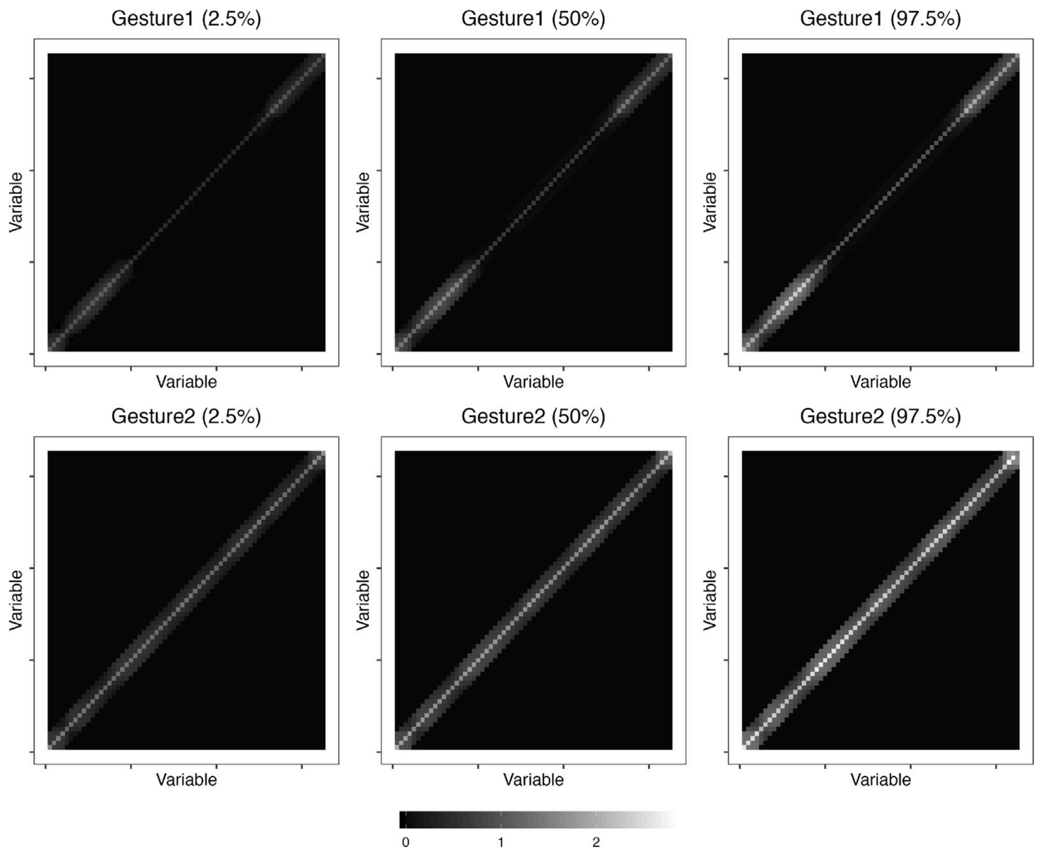


Fig. 5. Element-wise posterior uncertainty quantification for the banded covariance matrices of two gestures. Each row shows the 2.5%, 50% (median), and 97.5% quantiles.

computationally infeasible, such as high-dimensional genomics data analysis or covariance modeling in large-scale financial portfolios.

bspcov supports shrinkage priors and post-processed posterior methods, which enable stable and sparse estimation of high-dimensional covariance matrices. These features improve inference by providing both point estimates and uncertainty quantification, while reducing the computational burden compared to traditional Bayesian methods. The package has been applied in various domains such as cancer classification, time series analysis, and financial modeling, and is increasingly used in both research and educational settings.

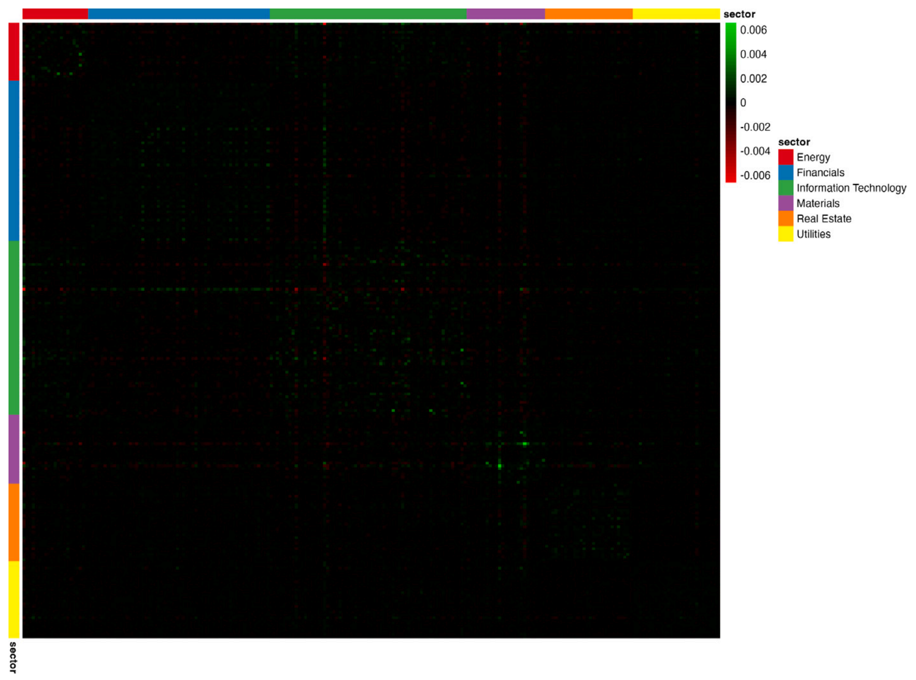


Fig. 6. The heatmap visualizes the estimated pairwise idiosyncratic covariances among stocks from six major industry sectors: Energy, Financials, Information Technology, Materials, Real Estate, and Utilities. To highlight within- and between-sector structures, stocks are ordered so that those belonging to the same sector appear in contiguous blocks. Colored bars along the axes indicate sector membership. Covariance values are color-coded from red (negative) through black (zero) to green (positive), with denser green blocks along the diagonal suggesting stronger within-sector dependencies.

While primarily developed for academic use, its scalable algorithms have attracted interest from practitioners in finance and data science. The software is freely available and continues to expand its user base.

5. Conclusions

Covariance matrices are a fundamental component in many statistical analyses, providing insights into understanding the dependence structure among different variables. In this paper, we have developed the `bspcov` R package for Bayesian inference of covariance matrices, particularly those with sparse or banded structures. Sparse matrices are those in which most elements are zero, and they frequently occur in high-dimensional datasets where many variables have no or negligible relationships. Banded covariance matrices, on the other hand, are those where significant values are confined to a diagonal band, encompassing the main diagonal and immediate off-diagonal elements. This structure is common in time series data or in datasets where the correlation between variables diminishes as the distance between them increases. Traditional methods of handling these matrices can be computationally intensive and less efficient. The `bspcov` package addresses this challenge by providing more efficient and effective tools for dealing with these types of matrices. In the package, the functions `bmspcov`, `sbmspcov`, and `thresPPP` are implemented to handle methods for sparse covariance matrices, while the function `bandPPP` is designed for banded structured covariance matrices.

The development of the `bspcov` package is a significant contribution to the field of statistical analysis, particularly in the age of complex and extensive datasets. By providing advanced tools for handling specific types of covariance matrix estimations, the package aids statisticians and data scientists in more accurately understanding the relationships between variables in large and complex datasets. This, in turn, can lead to more informed decision-making in various fields, from finance and economics to biology and social sciences. The ability to efficiently and effectively work with sparse and banded covariance matrices opens up new possibilities for data analysis, making the `bspcov` package a valuable addition to the R programming environment. The package is available to the R community though the Github repository.

It is worth noting, however, that the current implementation of the `bspcov` package is not designed to handle general, unstructured covariance matrices. Instead, it focuses on covariance matrices with sparse or banded structures, which are particularly relevant and beneficial in high-dimensional settings. While this design choice allows for efficient and stable estimation in such contexts, it may limit the applicability of the package in situations where the underlying covariance structure does not exhibit these specific patterns. We believe this clarification will help users better understand the scope and intended use of the package.

CRediT authorship contribution statement

Kyeongwon Lee: Writing – original draft, Software, Data curation. **Kyoungjae Lee:** Writing – review & editing, Writing – original draft, Methodology. **Seongil Jo:** Writing – review & editing, Software, Conceptualization. **Kwangmin Lee:** Writing – review & editing, Writing – original draft, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Kwangmin Lee was supported by the BK21 FOUR (Fostering Outstanding Universities for Research), South Korea (No. 5120200913674) funded by the Ministry of Education(MOE, Korea) and National Research Foundation of Korea (NRF), South Korea. Kyoungjae Lee was supported by the National Research Foundation of Korea (NRF), South Korea grant funded by the Korea government (MSIT) (RS-2023-00211974 and RS-2025-02216235). Seongil Jo was supported by INHA UNIVERSITY Research grant.

Appendix A. Details for the beta-mixture and screened beta-mixture shrinkage priors

Consider n independent samples $\mathbf{X}_n = (X_1, \dots, X_n)$ from the following p-dimensional normal distribution,

$$X_i \mid \Sigma \overset{iid}{\sim} N_p(0, \Sigma), \quad i = 1, \dots, n, \tag{A.1}$$

where $\Sigma \in \mathcal{C}_p$ and $\mathcal{C}_p$ is the set of all $p \times p$ positive-definite matrices. We assume that most of the off-diagonal entries of Σ are zero, i.e., the covariance matrix Σ is ℓ_0-sparse. We adopt an approach where, instead of directly declaring a prior distribution for $p \times p$ covariance matrices, we first specify a prior distribution for $p \times p$ symmetric matrices with positive diagonals and then constrain this to the space of $p \times p$ positive definite matrices.

Let

$$\begin{aligned} \pi^u(\sigma_{jk} \mid \psi_{jk}) &= N\left(\sigma_{jk} \,\middle|\, 0, \frac{\psi_{jk}}{1-\psi_{jk}}\tau_1^2\right), \\ \pi^u(\psi_{jk}) &= Beta(\psi_{jk} \mid a, b), \quad 1 \le j < k \le p, \\ \pi^u(\sigma_{jj}) &= Gamma(\sigma_{jj} \mid 1, \lambda/2), \quad j = 1, \dots, p, \end{aligned} \tag{A.2}$$

where τ_1, a, b, c and d are positive constants, $Beta(a, b)$ is the beta distribution with parameters a and b, and $Gamma(c, d)$ is the gamma distribution with shape parameter c and rate parameter d. We define a prior for $p \times p$ symmetric matrices with positive diagonals as

$$\pi^u(\Sigma) = \prod_{1 \le j < k \le p} \left\{ \int_0^1 \pi^u(\sigma_{jk} \mid \psi_{jk}) \pi^u(\psi_{jk}) d\psi_{jk} \right\} \prod_{j=1}^p \pi^u(\sigma_{jj}) I(\Sigma = \Sigma^T),$$

where u denotes an unconstrained prior. By default, we set the hyperparameters to $a = b = 1/2$ and $\lambda = 1$, although other choices are also possible. When the hyperparameters a and b are set to $a = b = 1/2$, the unconstrained prior distribution for σ_{jk} coincides with the horseshoe prior. Therefore, this choice plays an important role in reflecting the belief that the covariance matrix is sparse. If prior information on the variance of variables is available, it can be incorporated through λ. When the data are standardized, the default value $\lambda = 1$ may be a reasonable choice.

By constraining $\pi^u(\sigma)$ to the subspace of $p \times p$ positive definite matrices, say $\mathcal{U}(\tau)$, we finally define the prior distribution for covariance matrices as follows:

$$\pi(\Sigma) = \frac{\pi^u(\Sigma) I(\Sigma \in \mathcal{U}(\tau))}{\pi^u(\Sigma \in \mathcal{U}(\tau))}, \tag{A.3}$$

where

$$\mathcal{U}(\tau) = \left\{ \Sigma \in \mathcal{C}_p : \tau^{-1} \le \lambda_{\min}(\Sigma) \le \lambda_{\max}(\Sigma) \le \tau \right\}$$

for some constant $\tau > 1$. We refer to (A.3) as the BM prior [5]. The introduction of $\tau > 1$ to restrict the eigenvalues of covariance matrices is done for technical reasons, and, in practice, one may choose to ignore it and proceed with $\tau = \infty$.

The SBM prior [14] applies a screening procedure to the BM prior, fixing some off-diagonal entries to zero and performing inference only on the remaining ones. Specifically, for a given threshold $r > 0$, the set of screened indices for the covariance matrix is defined as

$$S_r = \left\{ (j, k) : 1 \le j < k \le p, \ |\hat{\rho}_{jk}| > r \right\},$$

where $\hat{\rho}_{jk}$ denotes the sample correlation between the jth and kth variables. In this procedure, the screening step retains only the off-diagonal entries whose absolute sample correlation coefficients exceed a given threshold $r > 0$. As described in the main text, the threshold r can be chosen using either an FNR-based approach or a correlation-based approach. We recommend that users adopt the default setting `cutoff = list(method = FNR, rho = 0.25, FNR = 0.05, nsimdata = 1000)`. This selects the threshold value such that the empirical FNR is 0.05, under the assumption that a correlation magnitude of 0.25 is considered meaningful.

The SBM prior requires the user to specify hyperparameters such as a, b, and τ_1^2, as well as the values for the cutoff method. However, the results in [14] indicate that when the true covariance matrix is sparse, the estimation performance of the SBM prior is not highly sensitive to the choice of hyperparameters. Furthermore, for the cutoff value, it was found that while it affects computation time, it has little impact on estimation performance.

Appendix B. Details for post-processed posteriors

To explain the arguments of `bandPPP` and `thresPPP`, we review the PPP methods introduced by [4,15]. The PPP method is a Bayesian method to estimate the covariance $\Sigma \in \mathcal{C}_p$, where $\mathcal{C}_p$ is the set of all $p \times p$ positive-definite matrices, in the multivariate normal model:

$$X_1, \dots, X_n \sim N_p(0, \Sigma), \text{ independently},$$

where n is the number of samples. The `bandPPP` (`thresPPP`) supposes that Σ is a banded (sparse) covariance matrix. The PPP methods consist of the initial posterior sampling and post-processing steps. Let $IW_p(A, \nu)$ be the initial prior distribution, where IW_p represents the inverse-Wishart distribution with scale matrix $A \in \mathcal{C}_p$ and scale parameter $\nu > 0$. Given $IW_p(A, \nu)$, the initial posterior samples are generated as follows:

$$\Sigma^{(1)}, \dots, \Sigma^{(N)} \sim IW_p(A + \sum_{i=1}^n X_i X_i^T, \nu + n),$$

where N is the number of posterior samples. In the second step, $\Sigma^{(1)}, \dots, \Sigma^{(N)}$ are transformed by a post-processing function. The `bandPPP` uses the banding operator defined as

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

where $B_k(B) = \{b_{ij} I(|i-j| \le k)\}$. The `thresPPP` uses the generalized thresholding operator defined as

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

where

$$(T_\gamma(B))_{ij} = \begin{cases} b_{ij}, & \text{if } i = j, \\ h_\gamma(b_{ij}), & \text{if } i \ne j, \end{cases}$$

and $h_\gamma(\cdot)$ is a generalized thresholding function satisfying the following conditions for all $z \in \mathbb{R}$:

(i) $h_\gamma(z) \le |z|$;
(ii) $h_\gamma(z) = 0$ for $|z| \le \gamma$;
(iii) $|h_\gamma(z) - z| \le \gamma$.

Appendix C. Advanced analysis with gene expression data

This section demonstrates the advanced capabilities of the `bspcov` package through comprehensive analysis of colon cancer gene expression data. We focus on sparse covariance matrix estimation using three primary methods: `bmspcov`, `sbmspcov`, and `thresPPP`. The analysis emphasizes uncertainty quantification, convergence diagnostics, and comparative performance evaluation.

C.1. Experimental setup

For the MCMC-based methods (`bmspcov` and `sbmspcov`), we employed the following settings:

- Burn-in period: 1000 iterations per chain

- MCMC samples: 2000 iterations per chain
- Number of chains: 4 (to enable convergence diagnostics)
- Monitoring focus: Diagonal elements around position 25 (rows/columns 24–26), chosen for their variation patterns

For `thresPPP`, we used 100 samples because the number of samples corresponds to the effective sample size. The diagonal elements were selected over off-diagonal elements for monitoring, as the latter frequently alternate between zero and non-zero states across chains, making convergence assessment challenging.

All experiments were conducted on a Laptop with Apple M4 Pro processor and 48 GB RAM, running macOS Sequoia 15.5. The computational analysis was performed using R version 4.4.3 on the aarch64-apple-darwin20 platform. The parallel processing capabilities of the `bspcov`, `rjags`, and `rstan` packages were utilized, taking advantage of the multi-core architecture for improved performance when running multiple MCMC chains simultaneously.

C.2. MCMC convergence diagnostics

For the MCMC-based methods, we examined convergence through multi-chain trace plots. Fig. C.7 displays the trace plots for the monitored diagonal elements (positions 24–26), with each chain shown in a different color.

The trace plots reveal varying degrees of autocorrelation across parameters. While some elements exhibit good mixing with random fluctuations around the posterior mean, others display stronger autocorrelation patterns, particularly evident in the slower exploration of the parameter space.

C.3. Quantitative convergence assessment

We computed comprehensive MCMC diagnostics using the enhanced `summary()` function for the monitored elements:

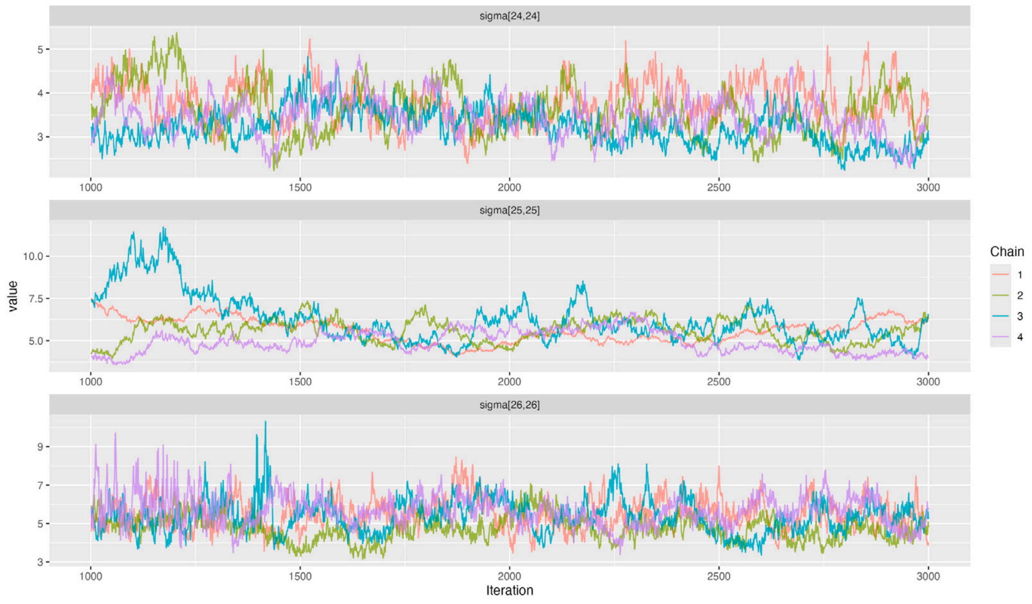
C.3.1. Results for bmspcov

```
# summary(foo.bmspcov, cols = 24:26, rows =
    24:26)

Number of chains: 4

MCMC Summary Statistics:

Iterations = 1001:3000
Thinning interval = 1
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for
    each variable,
   plus standard error of the mean:

               Mean     SD Naive SE Time-
    series SE
sigma[24,24] 3.512 0.5168 0.005778
    0.04463
sigma[25,25] 5.615 1.0403 0.011631
    0.17554
sigma[26,26] 5.328 0.8569 0.009580
    0.05775

2. Quantiles for each variable:

               2.5%   25%   50%   75% 97.5%
sigma[24,24] 2.593 3.135 3.479 3.842 4.610
sigma[25,25] 4.097 4.901 5.523 6.120 8.118
sigma[26,26] 3.817 4.727 5.280 5.867 7.114


MCMC Diagnostics:

Effective Sample Size (n_eff):
sigma[24,24] sigma[25,25] sigma[26,26]
       137.5          34.1         225.7

Potential Scale Reduction Factor (R-hat):
             Point est. Upper C.I.
sigma[24,24]      1.158      1.414
sigma[25,25]      1.122      1.324
sigma[26,26]      1.220      1.570

Warning: Some R-hat values > 1.1, indicating
    potential convergence issues.
```

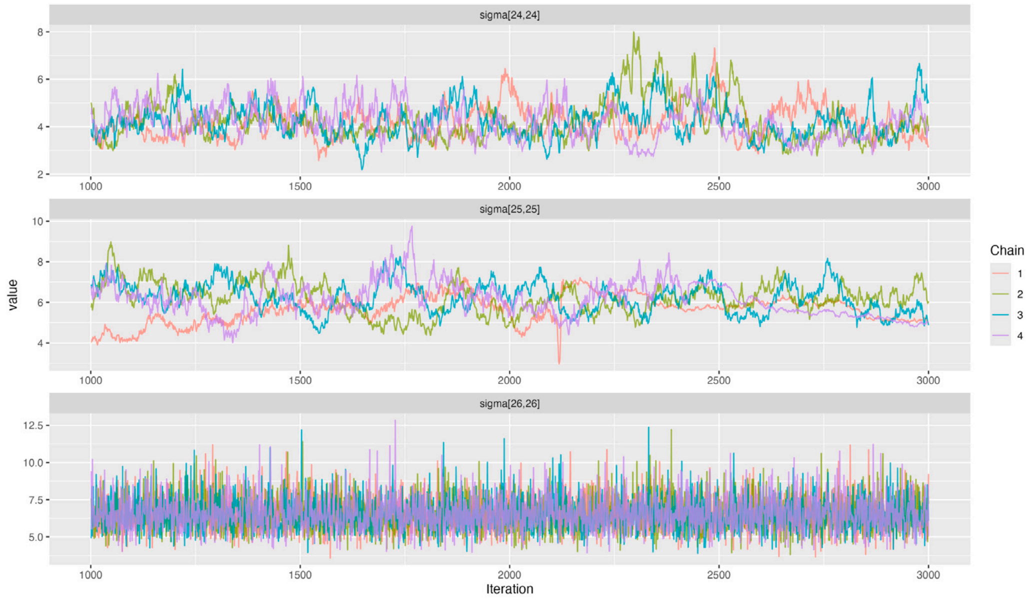
C.3.2. Results for sbmspcov

```
# summary(foo.sbmspcov, cols = 24:26, rows =
    24:26)

Number of chains: 4

MCMC Summary Statistics:

Iterations = 1001:3000
Thinning interval = 1
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for
    each variable,
   plus standard error of the mean:

               Mean     SD Naive SE Time-
    series SE
sigma[24,24] 4.212 0.7284 0.008143
    0.05704
sigma[25,25] 6.065 0.7856 0.008784
    0.08846
sigma[26,26] 6.537 1.1146 0.012462
    0.01229

2. Quantiles for each variable:

               2.5%   25%   50%   75% 97.5%
sigma[24,24] 3.073 3.684 4.117 4.617 5.916
sigma[25,25] 4.565 5.540 6.030 6.589 7.616
sigma[26,26] 4.679 5.762 6.410 7.209 8.979


MCMC Diagnostics:

Effective Sample Size (n_eff):
sigma[24,24] sigma[25,25] sigma[26,26]
       179.6          76.2        8239.5

Potential Scale Reduction Factor (R-hat):
             Point est. Upper C.I.
sigma[24,24]      1.015      1.030
sigma[25,25]      1.033      1.079
sigma[26,26]      1.000      1.001
```

The effective sample sizes around 100 for most parameters indicate strong autocorrelation in the Markov chains, consistent with the visual patterns observed in the trace plots. The `bmspcov` method shows some convergence concerns with R-hat values exceeding 1.1, while `sbmspcov` demonstrates better convergence properties with all R-hat values close to 1.0.



(a) `bmspcov`: Diagonal elements at positions 24-26



(b) `sbmspcov`: Diagonal elements at positions 24-26

Fig. C.7. Multi-chain trace plots for MCMC convergence diagnostics.

Table C.3
Performance comparison across methods: Mean (SD) of misclassification rate and computation time.

Method	Misclassification rate	Elapsed time (s)
thresPPP	**0.113 (0.044)**	**0.129 (0.003)**
sbmspcov	0.162 (0.056)	8.624 (0.204)
bmspcov	**0.113 (0.044)**	26.073 (0.586)
JAGS	**0.113 (0.044)**	64.705 (1.282)
Stan	**0.113 (0.044)**	452.220 (30.662)

C.4. Comparative performance analysis

We conducted 5-fold cross-validation experiments comparing our methods with established Bayesian MCMC packages (JAGS and Stan) using inverse-Wishart priors. Table C.3 summarizes the results:

The results demonstrate that `thresPPP` achieves competitive accuracy with remarkably fast computation times (sub-second), while the MCMC-based methods in `bspcov` package significantly outperform traditional implementations in computational efficiency. The `sbmspcov` method shows slightly higher misclassification rates due to its aggressive screening procedure but offers substantial speed improvements over `bmspcov`.

References

[1] Cai Tony, Liu Weidong. Adaptive thresholding for sparse covariance matrix estimation. J Amer Statist Assoc 2011;106(494):672–84.

[2] Cai Tony, Zhou Harrison. Optimal rates of convergence for sparse covariance matrix estimation. Ann Statist 2012;40(5):2389–420.

[3] Lee Kwangmin, Lee Kyoungjae, Lee Jaeyong. Estimation of conditional mean operator under the bandable covariance structure. Electron J Stat 2022;16(1):1253–302.

[4] Lee Kwangmin, Lee Kyoungjae, Lee Jaeyong. Post-processed posteriors for banded covariances. Bayesian Anal 2023;18(3):1017–40.

[5] Lee Kyoungjae, Jo Seongil, Lee Jaeyong. The beta-mixture shrinkage prior for sparse covariances with near-minimax posterior convergence rate. J Multivariate Anal 2022;192:105067.

[6] Friedman Jerome, Hastie Trevor, Tibshirani Rob. glasso: Graphical lasso – estimation of Gaussian graphical models. 2022, R package version 1.11, URL https://cran.r-project.org/package=glasso.

[7] Zhao Tuo, Liu Han, Roeder Kathryn, Lafferty John, Wasserman Larry. The huge package for high-dimensional undirected graph estimation in R. J Mach Learn Res 2012;13:1059–62.

[8] Li Xingguo, Zhao Tuo, Yuan Xiaoming, Liu Han. The `flare` package for high-dimensional linear regression and precision matrix estimation in R. J Mach Learn Res 2015;16:553–7, R package.

[9] Lee Kyoungjae, Lin Lizhen, You Kisung. CovTools: Statistical tools for covariance analysis. 2021, R package version 0.5.4, URL https://cran.r-project.org/package=CovTools.

[10] Peeters Carel FW, Bilgrau Anders E, van Wieringen Wessel N. rags2ridges: A one-stop-ℓ_2-shop for graphical modeling of high-dimensional precision matrices. J Stat Softw 2022;102(4):1–32. http://dx.doi.org/10.18637/jss.v102.i04.

[11] Bien Jacob, Tibshirani Robert. Spcov: Sparse estimation of a covariance matrix. 2022, R package version 1.3, URL https://cran.r-project.org/package=spcov.

[12] Mohammadi Reza, Wit Ernst C. BDgraph: An R package for Bayesian structure learning in graphical models. J Stat Softw 2019;89(3):1–30.

[13] Williams Donald R, Mulder Joris. BGGM: Bayesian Gaussian graphical models in R. J Open Source Softw 2020;5(51):2111. http://dx.doi.org/10.21105/joss.02111.

[14] Lee Kyoungjae, Lee Kyeongwon, Jo Seongil, Lee Jaeyong. Scalable and optimal Bayesian inference for sparse covariance matrices via screened beta-mixture prior. Bayesian Anal 2024. http://dx.doi.org/10.1214/24-BA1495, Advance publication.

[15] Lee Kwangmin, Lee Jaeyong. Post-processed posteriors for sparse covariances. J Econometrics 2023;236(1):105475.

[16] Wang Hao. Scaling it up: stochastic search structure learning in graphical models. Bayesian Anal 2015;10(2):351–77.

[17] Ly Alexander, Verhagen Josine, Wagenmakers Eric-Jan. Harold Jeffreys's default Bayes factor hypothesis tests: Explanation, extension, and application in psychology. J Math Psych 2016;72:19–32.

[18] Gelman Andrew, Rubin Donald B. Inference from iterative simulation using multiple sequences. Statist Sci 1992;7(4):457–72.

[19] Rothman Adam J, Bickel Peter J, Levina Elizaveta, Zhu Ji. Sparse permutation invariant covariance estimation. Electron J Stat 2008;2:494–515.

[20] Bagnall Anthony, Dau Hoang Anh, Lines Jason, Flynn Michael, Large James, Bostrom Aaron, et al. The UEA multivariate time series classification archive, 2018. 2018, arXiv preprint arXiv:1811.00075.

[21] State Street Global Advisors. 2023. URL https://www.ssga.com/. [Accessed 8 December 2023].

[22] Chamberlain Gary, Rothschild Michael. Arbitrage, factor structure, and mean-variance analysis on large asset markets. Econometrica 1983;51(5):1281–304.