

# Package: varband (via r-universe)

October 1, 2026

**Type** Package  
**Title** Variable Banding of Large Precision Matrices  
**Version** 0.9.1  
**Description** Implementation of the variable banding procedure for modeling local dependence and estimating precision matrices that is introduced in Yu & Bien (2016) and is available at <https://arxiv.org/abs/1604.07451>.  
**License** GPL-3  
**LazyData** TRUE  
**Suggests** knitr, rmarkdown  
**VignetteBuilder** knitr  
**RoxygenNote** 5.0.1  
**URL** <http://github.com/hugogogo/varband>  
**BugReports** <http://github.com/hugogogo/varband/issues>  
**Collate** 'model\_gen.R' 'refit.R' 'varband\_cv.R' 'varband\_path.R' 'RcppExports.R' 'misc.R' 'utils.R'  
**LinkingTo** Rcpp, RcppArmadillo  
**Imports** Rcpp, stats, graphics  
**Repository** <https://hugogogo.r-universe.dev>  
**Date/Publication** 2017-12-27 04:44:06 UTC  
**RemoteUrl** <https://github.com/hugogogo/varband>  
**RemoteRef** HEAD  
**RemoteSha** aca7e8497ac9d881b50a4e0655ad83547bf2f445

## Contents

|                          |   |
|--------------------------|---|
| ar_gen . . . . .         | 2 |
| block_diag_gen . . . . . | 2 |
| matimage . . . . .       | 3 |

2

block\_diag\_gen

sample\_gen . . . . . 3

varband . . . . . 4

varband\_cv . . . . . 5

varband\_gen . . . . . 7

varband\_path . . . . . 8

Index 9

---

|        |                                          |
|--------|------------------------------------------|
| ar_gen | <i>Generate an autoregressive model.</i> |
|--------|------------------------------------------|

---

**Description**

Generate lower triangular matrix with strict bandwidth. See, e.g., Model 1 in the paper.

**Usage**

```
ar_gen(p, phi_vec)
```

**Arguments**

- p                    the dimension of L
- phi\_vec            a K-dimensional vector for off-diagonal values

**Value**

a p-by-p strictly banded lower triangular matrix

**Examples**

```
true_ar <- ar_gen(p = 50, phi = c(0.5, -0.4, 0.1))
```

---

|                |                                                       |
|----------------|-------------------------------------------------------|
| block_diag_gen | <i>Generate a model with block-diagonal structure</i> |
|----------------|-------------------------------------------------------|

---

**Description**

Generate a model with block-diagonal structure

**Usage**

```
block_diag_gen(p)
```

**Arguments**

- p                    the dimension of L

**Value**

a p-by-p lower triangular matrix with block-diagonal structure from p/4-th row to 3p/4-th row

**Examples**

```
set.seed(123)
true_L_block_diag <- block_diag_gen(p = 50)
```

---

|          |                                                     |
|----------|-----------------------------------------------------|
| matimage | <i>Plot the sparsity pattern of a square matrix</i> |
|----------|-----------------------------------------------------|

---

**Description**

Black, white and gray stand for positive, zero and negative respectively

**Usage**

```
matimage(Mat, main = NULL)
```

**Arguments**

|      |                   |
|------|-------------------|
| Mat  | A matrix to plot. |
| main | A plot title.     |

**Examples**

```
set.seed(123)
p <- 50
n <- 50
phi <- 0.4
true <- varband_gen(p = p, block = 5)
matimage(true)
```

---

|            |                                 |
|------------|---------------------------------|
| sample_gen | <i>Generate random samples.</i> |
|------------|---------------------------------|

---

**Description**

Generate n random samples from multivariate Gaussian distribution  $N(0, (L^{TL})^{-1})$

**Usage**

```
sample_gen(L, n)
```

**Arguments**

L                      p-dimensional inverse Cholesky factor of true covariance matrix.  
n                        number of samples to generate.

**Value**

returns a n-by-p matrix with each row a random sample generated.

**Examples**

```
set.seed(123)
true <- varband_gen(p = 50, block = 5)
x <- sample_gen(L = true, n = 100)
```

---

|         |                                                                                                        |
|---------|--------------------------------------------------------------------------------------------------------|
| varband | <i>Compute the varband estimate for a fixed tuning parameter value with different penalty options.</i> |
|---------|--------------------------------------------------------------------------------------------------------|

---

**Description**

Solves the main optimization problem in Yu & Bien (2017):

$$\min_L -2 \sum_{r=1}^p L_{rr} + \text{tr}(SLL^T) + \text{lam} * \sum_{r=2}^p P_r(L_{r.})$$

where

$$P_r(L_{r.}) = \sum_{\ell=2}^{r-1} \left( \sum_{m=1}^{\ell} w_{\ell m}^2 L_{rm}^2 \right)^{1/2}$$

or

$$P_r(L_{r.}) = \sum_{\ell=1}^{r-1} |L_{r\ell}|$$

**Usage**

```
varband(S, lambda, init, K = -1L, w = FALSE, lasso = FALSE)
```

**Arguments**

S                      The sample covariance matrix  
lambda                Non-negative tuning parameter. Controls sparsity level.  
init                    Initial estimate of L. Default is a closed-form diagonal estimate of L.  
K                       Integer between 0 and p - 1 (default), indicating the maximum bandwidth in the resulting estimate. A small value of K will result in a sparse estimate and small computing time.

|       |                                                                                                                                                                                         |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| w     | Logical. Should we use weighted version of the penalty or not? If TRUE, we use general weight. If FALSE, use unweighted penalty. Default is FALSE.                                      |
| lasso | Logical. Should we use l1 penalty instead of hierarchical group lasso penalty? Note that by using l1 penalty, we lose the banded structure in the resulting estimate. Default is FALSE. |

### Details

The function decomposes into  $p$  independent row problems, each of which is solved by an ADMM algorithm. see paper for more explanation.

### Value

Returns the variable banding estimate of  $L$ , at most  $K$  banded, where  $L^{TL} = \Omega$ .

### See Also

[varband\\_path](#) [varband\\_cv](#)

### Examples

```
set.seed(123)
n <- 50
true <- varband_gen(p = 50, block = 5)
x <- sample_gen(L = true, n = n)
S <- crossprod(scale(x, center = TRUE, scale = FALSE)) / n
init <- diag(1/sqrt(diag(S)))
# unweighted estimate
L_unweighted <- varband(S, lambda = 0.1, init, w = FALSE)
# at most 10-banded unweighted estimate
L_unweighted_K10 <- varband(S, lambda = 0.1, init, w = FALSE, K = 10)
# weighted estimate
L_weighted <- varband(S, lambda = 0.1, init, w = TRUE)
# lasso estimate
L_lasso <- varband(S, lambda = 0.1, init, w = TRUE, lasso = TRUE)
```

---

varband\_cv

---

*Perform nfolds-cross validation*


---

### Description

Select tuning parameter by cross validation according to the likelihood on testing data, with and without refitting.

### Usage

```
varband_cv(x, w = FALSE, lasso = FALSE, K = -1, lamlist = NULL,
  nlam = 60, flmin = 0.01, folds = NULL, nfolds = 5)
```

## Arguments

|                               |                                                                                                                                                                                                                                                                                                                           |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>                | A n-by-p sample matrix, each row is an observation of the p-dim random vector.                                                                                                                                                                                                                                            |
| <code>w</code>                | Logical. Should we use weighted version of the penalty or not? If TRUE, we use general weight. If FALSE, use unweighted penalty. Default is FALSE.                                                                                                                                                                        |
| <code>lasso</code>            | Logical. Should we use l1 penalty instead of hierarchical group lasso penalty? Note that by using l1 penalty, we lose the banded structure in the resulting estimate. And when using l1 penalty, the becomes CSCS (Convex Sparse Cholesky Selection) introduced in Khare et al. (2016). Default value for lasso is FALSE. |
| <code>K</code>                | Integer between 0 and $p - 1$ (default), indicating the maximum bandwidth in the resulting estimate. A small value of K will result in a sparse estimate and small computing time.                                                                                                                                        |
| <code>lamlist</code>          | A list of non-negative tuning parameters $\lambda$ .                                                                                                                                                                                                                                                                      |
| <code>n<sub>lam</sub></code>  | If lamlist is not provided, create a lamlist with length n <sub>lam</sub> . Default is 60.                                                                                                                                                                                                                                |
| <code>fl<sub>min</sub></code> | If lamlist is not provided, create a lamlist with ratio of the smallest and largest $\lambda$ in the list equal to fl <sub>min</sub> . Default is 0.01.                                                                                                                                                                   |
| <code>folds</code>            | Folds used in cross-validation                                                                                                                                                                                                                                                                                            |
| <code>nfolds</code>           | If folds are not provided, create folds of size nfolds.                                                                                                                                                                                                                                                                   |

## Value

A list object containing

**errs\_fit:** A n<sub>lam</sub>-by-nfolds matrix of negative Gaussian log-likelihood values on the CV test data sets. `errs[i, j]` is negative Gaussian log-likelihood values incurred in using `lamlist[i]` on fold `j`.

**errs\_refit:** A n<sub>lam</sub>-by-nfolds matrix of negative Gaussian log-likelihood values of the refitting.

**folds:** Folds used in cross validation.

**lamlist:**  $\lambda$  grid used in cross validation.

**ibest\_fit:** index of lamlist minimizing CV negative Gaussian log-likelihood.

**ibest\_refit:** index of lamlist minimizing refitting CV negative Gaussian log-likelihood.

**ilse\_fit:** Selected value of  $\lambda$  using the one-standard-error rule.

**ilse\_refit:** Selected value of  $\lambda$  of the refitting process using the one-standard-error rule.

**L\_fit:** Estimate of  $L$  corresponding to `ibest_fit`.

**L\_refit:** Refitted estimate of  $L$  corresponding to `ibest_refit`.

## See Also

[varband](#) [varband\\_path](#)

**Examples**

```

set.seed(123)
p <- 50
n <- 50
true <- varband_gen(p = p, block = 5)
x <- sample_gen(L = true, n = n)
res_cv <- varband_cv(x = x, w = FALSE, nlam = 40, flmin = 0.03)

```

---

|             |                                                  |
|-------------|--------------------------------------------------|
| varband_gen | <i>Generate a model with variable bandwidth.</i> |
|-------------|--------------------------------------------------|

---

**Description**

Generate lower triangular matrix with variable bandwidth. See, e.g., Model 2 and 3 in the paper.

**Usage**

```
varband_gen(p, block = 10)
```

**Arguments**

|       |                                                                                     |
|-------|-------------------------------------------------------------------------------------|
| p     | the dimension of L                                                                  |
| block | the number of block diagonal structures in the resulting model, assumed to divide p |

**Value**

a p-by-p lower triangular matrix with variable bandwidth

**Examples**

```

set.seed(123)
# small block size (big number of blocks)
true_small <- varband_gen(p = 50, block = 10)
# large block size (small number of blocks)
true_large <- varband_gen(p = 50, block = 2)

```

---

|              |                                                               |
|--------------|---------------------------------------------------------------|
| varband_path | <i>Solve main optimization problem along a path of lambda</i> |
|--------------|---------------------------------------------------------------|

---

## Description

Compute the varband estimates along a path of tuning parameter values.

## Usage

```
varband_path(S, w = FALSE, lasso = FALSE, K = -1, lamlist = NULL,
             nlam = 60, flmin = 0.01)
```

## Arguments

|                |                                                                                                                                                                                                                                                                                                                           |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>S</b>       | The sample covariance matrix                                                                                                                                                                                                                                                                                              |
| <b>w</b>       | Logical. Should we use weighted version of the penalty or not? If TRUE, we use general weight. If FALSE, use unweighted penalty. Default is FALSE.                                                                                                                                                                        |
| <b>lasso</b>   | Logical. Should we use l1 penalty instead of hierarchical group lasso penalty? Note that by using l1 penalty, we lose the banded structure in the resulting estimate. And when using l1 penalty, the becomes CSCS (Convex Sparse Cholesky Selection) introduced in Khare et al. (2016). Default value for lasso is FALSE. |
| <b>K</b>       | Integer between 0 and $p - 1$ (default), indicating the maximum bandwidth in the resulting estimate. A small value of K will result in a sparse estimate and small computing time.                                                                                                                                        |
| <b>lamlist</b> | A list of non-negative tuning parameters lambda.                                                                                                                                                                                                                                                                          |
| <b>nlam</b>    | If lamlist is not provided, create a lamlist with length node. Default is 60.                                                                                                                                                                                                                                             |
| <b>flmin</b>   | if lamlist is not provided, create a lamlist with ratio of the smallest and largest lambda in the list. Default is 0.01.                                                                                                                                                                                                  |

## Value

A list object containing

**path:** A array of dim  $(p, p, nlam)$  of estimates of L

**lamlist:** a grid values of tuning parameters

## See Also

[varband](#) [varband\\_cv](#)

## Examples

```
set.seed(123)
n <- 50
true <- varband_gen(p = 50, block = 5)
x <- sample_gen(L = true, n = n)
S <- crossprod(scale(x, center = TRUE, scale = FALSE))/n
path_res <- varband_path(S = S, w = FALSE, nlam = 40, flmin = 0.03)
```

# Index

ar\_gen, [2](#)

block\_diag\_gen, [2](#)

matimage, [3](#)

sample\_gen, [3](#)

varband, [4](#), [6](#), [8](#)

varband\_cv, [5](#), [5](#), [8](#)

varband\_gen, [7](#)

varband\_path, [5](#), [6](#), [8](#)