

# Package ‘mglasso’

September 8, 2022

**Type** Package

**Title** Multiscale Graphical Lasso

**Version** 0.1.2

**Description** Inference of Multiscale graphical models with neighborhood selection approach. The method is based on solving a convex optimization problem combining a Lasso and fused-group Lasso penalties. This allows to infer simultaneously a conditional independence graph and a clustering partition. The optimization is based on the Continuation with Nesterov smoothing in a Shrinkage-Thresholding Algorithm solver (Hadj-Selem et al. 2018) <doi:10.1109/TMI.2018.2829802> implemented in python.

**License** MIT + file LICENSE

**Imports** corpcor, ggplot2, ggrepel, gridExtra, Matrix, methods, R.utils, reticulate (>= 1.25), rstudioapi

**Suggests** knitr, mvtnorm, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**ByteCompile** true

**Config/reticulate** list( packages = list( list(package = ``scipy``, version = ``1.7.1``), list(package = ``numpy``, version = ``1.22.4``), list(package = ``matplotlib``), list(package = ``scikit-learn``), list(package = ``six``), list(package = ``pylearn-parsimony``, version = ``0.3.1``, pip = TRUE) ) )

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**URL** <https://desanou.github.io/mglasso/>

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Edmond Sanou [aut, cre],  
Tung Le [ctb],  
Christophe Ambroise [ths],  
Geneviève Robin [ths]

**Maintainer** Edmond Sanou <doedmond.sanou@univ-evry.fr>

**Repository** CRAN

**Date/Publication** 2022-09-08 13:12:55 UTC

## R topics documented:

adj_mat . . . . .	2
beta_ols . . . . .	3
beta_to_vector . . . . .	3
conesta . . . . .	4
cost . . . . .	5
dist_beta . . . . .	6
fun_lines . . . . .	6
image_sparse . . . . .	7
install_pylearn_parsimony . . . . .	8
merge_clusters . . . . .	9
mglasso . . . . .	9
plot_clusterpath . . . . .	11
plot_mglasso . . . . .	12
precision_to_regression . . . . .	12
symmetrize . . . . .	13

**Index** **14**

---

adj_mat	<i>Adjacency matrix</i>
---------	-------------------------

---

### Description

Adjacency matrix

### Usage

```
adj_mat(mat, sym_rule = "and")
```

### Arguments

mat	matrix of regression coefficients
sym_rule	symmetrization rule, either AND or OR

### Value

adjacency matrix

---

beta_ols	<i>Initialize regression matrix</i>
----------	-------------------------------------

---

**Description**

Initialize regression matrix

**Usage**

```
beta_ols(X)
```

**Arguments**

X	data
---	------

**Value**

A zero-diagonal matrix of regression vectors.

---

beta_to_vector	<i>Transform a matrix of regression coefficients to vector removing the diagonal</i>
----------------	--

---

**Description**

Transform a matrix of regression coefficients to vector removing the diagonal

**Usage**

```
beta_to_vector(beta_mat)
```

**Arguments**

beta_mat	matrix of regressions vectors
----------	-------------------------------

**Value**

A numeric vector of all regression coefficients.

conesta

*CONESTA solver.***Description**

Solve the MGLasso optimization problem using CONESTA algorithm. Interface to the `pylearn.parsimony` python library.

**Usage**

```
conesta(
  X,
  lam1,
  lam2,
  beta_warm = c(0),
  type_ = "initial",
  W_ = NULL,
  mean_ = FALSE,
  max_iter_ = 10000,
  prec_ = 0.01
)
```

**Arguments**

<code>X</code>	Data matrix $n \times p$ .
<code>lam1</code>	Sparsity penalty.
<code>lam2</code>	Total variation penalty.
<code>beta_warm</code>	Warm initialization vector.
<code>type_</code>	Character scalar. By default set to initial version which doesn't use weights
<code>W_</code>	Weights matrix for total variation penalties.
<code>mean_</code>	Logical scalar. If TRUE weights the optimization function by the inverse of sample size.
<code>max_iter_</code>	Numeric scalar. Maximum number of iterations.
<code>prec_</code>	Numeric scalar. Tolerance for the stopping criterion (duality gap).

**Details**

*Continuation with NEsterov smoothing in a Shrinkage-Thresholding Algorithm* (CONESTA, Hadj-Seleem et al. 2018) [doi:10.1109/TMI.2018.2829802](https://doi.org/10.1109/TMI.2018.2829802) is an algorithm design for solving optimization problems including group-wise penalties. This function is an interface with the python solver. The MGLasso problem is first reformulated in a problem of the form

$$\operatorname{argmin} 1/2 \|Y - \tilde{X} \tilde{\beta}\|_2^2 + \lambda_1 \|\tilde{\beta}\|_1 + \lambda_2 \sum_{i < j} \|A_{ij} \tilde{\beta}\|_2$$

where vector  $Y$  is the vectorized form of matrix  $X$ .

**Value**

Numeric matrix of size  $p \times p$ . Line  $k$  of the matrix represents the coefficients obtained from the L1-L2 penalized regression of variable  $k$  on the others.

**See Also**

[mglasso\(\)](#) for the MGLasso model estimation.

**Examples**

```
## Not run: # because of installation of external packages during checks
mglasso::install_pylearn_parsimony(envname = "rmglasso", method = "conda")
reticulate::use_condaenv("rmglasso", required = TRUE)
reticulate::py_config()
n = 30
K = 2
p = 4
rho = 0.85
blocs <- list()
for (j in 1:K) {
  bloc <- matrix(rho, nrow = p/K, ncol = p/K)
  for(i in 1:(p/K)) { bloc[i,i] <- 1 }
  blocs[[j]] <- bloc
}

mat.covariance <- Matrix::bdiag(blocs)
mat.covariance
set.seed(11)
X <- mvtnorm::rmvnorm(n, mean = rep(0,p), sigma = as.matrix(mat.covariance))
X <- scale(X)
res <- conesta(X, 0.1, 0.1)

## End(Not run)
```

---

 cost

 Mglasso *cost function*


---

**Description**

cost computes the cost function of Mglasso method.

**Usage**

```
cost(beta, x, lambda1 = 0, lambda2 = 0)
```

**Arguments**

beta	p by p numeric matrix. In rows, regression vectors coefficients after node-wise regression. $\text{diag}(\text{beta}) = 0$ .
x	n by p numeric matrix. Data with variables in columns.
lambda1	numeric scalar. Lasso penalization parameter.
lambda2	numeric scalar. Fused-group Lasso penalization parameter.

**Value**

numeric scalar. The cost.

---

dist_beta	<i>Compute distance matrix between regression vectors</i>
-----------	---

---

**Description**

Compute distance matrix between regression vectors

**Usage**

```
dist_beta(beta, distance = "euclidean")
```

**Arguments**

beta	matrix of regression vectors
distance	euclidean or relative distance

**Value**

A numeric matrix of distances.

---

fun_lines	<i>weighted sum/difference of two regression vectors</i>
-----------	--

---

**Description**

fun\_lines applies function fun to regression vectors while reordering the coefficients, such that the j-th coefficient in  $\text{beta}[j, ]$  is permuted with the i-th coefficient.

**Usage**

```
fun_lines(i, j, beta, fun = `-\`, ni = 1, nj = 1)
```

**Arguments**

i	integer scalar. Index of the first vector.
j	integer scalar. Index of the second vector.
beta	p by p numeric matrix. In rows, regression vectors coefficients after node-wise regression. $\text{diag}(\text{beta}) = 0$ .
fun	function. Applied on lines.
ni	integer scalar. Weight for vector i.
nj	integer scalar. Weight for vector j.

**Value**

numeric vector

**Examples**

```
beta <- matrix(round(rnorm(9),2), ncol = 3)
diag(beta) <- 0
beta
fun_lines(1, 2, beta)
fun_lines(2, 1, beta)
```

---

image\_sparse

*Plot the image of a matrix*

---

**Description**

Plot the image of a matrix

**Usage**

```
image_sparse(matrix, main_ = "", sub_ = "", col_names = FALSE)
```

**Arguments**

matrix	matrix of regression coefficients
main_	title
sub_	subtitle
col_names	columns names

**Value**

No return value.

---

```
install_pylearn_parsimony
```

*Install the python library pylearn-parsimony and other required libraries*

---

### Description

pylearn-parsimony contains the solver CONESTA used for the mglasso problem and is available on github at <https://github.com/neurospin/pylearn-parsimony> It is advised to use a python version " $\geq 3.7, < 3.10$ ". Indeed, the latest version of scipy under which mglasso was developed is scipy 1.7.1 which is based on python " $\geq 3.7, < 3.10$ ". In turn, this version of scipy can only be associated with a version of numpy " $\geq 1.16.5, < 1.23.0$ "

### Usage

```
install_pylearn_parsimony(
  method = c("auto", "virtualenv", "conda"),
  conda = "auto",
  extra_pack = c("scipy == 1.7.1", "scikit-learn", "numpy == 1.22.4", "six",
    "matplotlib"),
  python_version = "3.8",
  restart_session = TRUE,
  envname = NULL,
  ...
)
```

### Arguments

method	Installation method. By default, "auto" automatically finds a method that will work in the local environment. Change the default to force a specific installation method. Note that the "virtualenv" method is not available on Windows.
conda	The path to a conda executable. Use "auto" to allow reticulate to automatically find an appropriate conda binary. See <b>Finding Conda</b> and <a href="#">conda_binary()</a> for more details.
extra_pack	Character vector. Extra-packages to be installed.
python_version	The requested Python version. Ignored when attempting to install with a Python virtual environment.
restart_session	Restart R session after installing (note this will only occur within RStudio)
envname	The name, or full path, of the environment in which Python packages are to be installed. When NULL (the default), the active environment as set by the RETICULATE_PYTHON_ENV variable will be used; if that is unset, then the r-reticulate environment will be used.
...	additional arguments passed to <a href="#">reticulate::py_install()</a>

**Value**

No return value.

---

merge_clusters	<i>compute clusters partition from pairs of variables to merge</i>
----------------	--

---

**Description**

compute clusters partition from pairs of variables to merge

**Usage**

```
merge_clusters(pairs_to_merge, clusters)
```

**Arguments**

pairs\_to\_merge table of the indices of variables to be merge  
 clusters numeric vector. By default 1:p where p is the number of variables

**Value**

A numeric vector.

---

mglasso	<i>Inference of Multiscale Gaussian Graphical Model.</i>
---------	--

---

**Description**

Cluster variables using L2 fusion penalty and simultaneously estimates a gaussian graphical model structure with the addition of L1 sparsity penalty.

**Usage**

```
mglasso(
  x,
  lambda1 = 0,
  fuse_thresh = 0.001,
  maxit = NULL,
  distance = c("euclidean", "relative"),
  lambda2_start = 1e-04,
  lambda2_factor = 1.5,
  precision = 0.01,
  weights_ = NULL,
  type = c("initial"),
  compact = TRUE,
  verbose = FALSE
)
```

**Arguments**

x	Numeric matrix ( $n \times p$ ). Multivariate normal sample with $n$ independent observations.
lambda1	Positive numeric scalar. Lasso penalty.
fuse_thresh	Positive numeric scalar. Threshold for clusters fusion.
maxit	Integer scalar. Maximum number of iterations.
distance	Character. Distance between regression vectors with permutation on symmetric coefficients.
lambda2_start	Numeric scalar. Starting value for fused-group Lasso penalty (clustering penalty).
lambda2_factor	Numeric scalar. Step used to update fused-group Lasso penalty in a multiplicative way..
precision	Tolerance for the stopping criterion (duality gap).
weights_	Matrix of weights.
type	If "initial" use classical version of <b>MGLasso</b> without weights.
compact	Logical scalar. If TRUE, only save results when previous clusters are different from current.
verbose	Logical scalar. Print trace. Default value is FALSE.

**Details**

Estimates a gaussian graphical model structure while hierarchically grouping variables by optimizing a pseudo-likelihood function combining Lasso and fuse-group Lasso penalties. The problem is solved via the *COntinuation with NEsteroV smoothing in a Shrinkage-Thresholding Algorithm* (Hadj-Selem et al. 2018). Varying the fusion penalty  $\lambda_2$  in a multiplicative fashion allow to uncover a seemingly hierarchical clustering structure. For  $\lambda_2 = 0$ , the approach is theoretically equivalent to the Meinshausen-Bühlmann (2006) *neighborhood selection* as resuming to the optimization of *pseudo-likelihood* function with  $\ell_1$  penalty (Rocha et al., 2008). The algorithm stops when all the variables have merged into one cluster. The criterion used to merge clusters is the  $\ell_2$ -norm distance between regression vectors.

For each iteration of the algorithm, the following function is optimized :

$$1/2 \sum_{i=1}^p \|X^i - X^{-i} \beta^i\|_2^2 + \lambda_1 \sum_{i=1}^p \|\beta^i\|_1 + \lambda_2 \sum_{i < j} \|\beta^i - \tau_{ij}(\beta^j)\|_2.$$

where  $\beta^i$  is the vector of coefficients obtained after regression  $X^i$  on the others and  $\tau_{ij}$  is a permutation exchanging  $\beta_j^i$  with  $\beta_i^j$ .

**Value**

A list-like object of class mglasso is returned.

out	List of lists. Each element of the list corresponds to a clustering level. An element named levelk contains the regression matrix beta and clusters vector clusters for a clustering in k clusters. When compact = TRUE out has as many elements as the number of unique partitions. When set to FALSE, the function returns as many items as the the range of values taken by lambda2.
l1	the sparsity penalty lambda1 used in the problem solving.

**See Also**

[conesta\(\)](#) for the problem solver, [plot\\_mglasso\(\)](#) for plotting the output of mglasso.

**Examples**

```
## Not run:
reticulate::use_condaenv("rmglasso", required = TRUE)
n = 50
K = 3
p = 9
rho = 0.85
blocs <- list()
for (j in 1:K) {
  bloc <- matrix(rho, nrow = p/K, ncol = p/K)
  for(i in 1:(p/K)) { bloc[i,i] <- 1 }
  blocs[[j]] <- bloc
}

mat.covariance <- Matrix::bdiag(blocs)
mat.covariance

set.seed(11)
X <- mvtnorm::rmvnorm(n, mean = rep(0,p), sigma = as.matrix(mat.covariance))
X <- scale(X)

res <- mglasso(X, 0.1, lambda2_start = 0.1)
res$out[[1]]$clusters
res$out[[1]]$beta

## End(Not run)
```

---

plot\_clusterpath      *Plot MGLasso Clusterpath*

---

**Description**

Plot MGLasso Clusterpath

**Usage**

```
plot_clusterpath(X, mglasso_res, colnames_ = NULL)
```

**Arguments**

X	numeric matrix
mglasso_res	object of class mglasso
colnames_	columns labels

**Details**

This function plot the clustering path of mglasso method on the 2 principal components axis of X. As the centroids matrices are not of the same dimension as X, we choose to plot the predicted X matrix path.

**Value**

no return value.

---

plot_mglasso	<i>Plot mglasso function output.</i>
--------------	--------------------------------------

---

**Description**

Plot the object returned by the mglasso function.

**Usage**

```
plot_mglasso(mglasso_, levels_ = NULL)
```

**Arguments**

mglasso_	Object of class mglasso.
levels_	Character vector. Selected levels for which estimated matrices will be plot. If NULL plot all levels.

**Value**

No return value.

---

precision_to_regression	<i>Compute precision matrix from regression vectors</i>
-------------------------	---

---

**Description**

Compute precision matrix from regression vectors

**Usage**

```
precision_to_regression(K)
```

**Arguments**

K	precision matrix
---	------------------

**Value**

A numeric matrix.

---

symmetrize	<i>Apply symmetrization on estimated graph</i>
------------	--

---

**Description**

Apply symmetrization on estimated graph

**Usage**

```
symmetrize(mat, rule = "and")
```

**Arguments**

mat	graph or precision matrix
rule	"and" or "or" rule

**Value**

A numeric matrix.

# Index

adj\_mat, [2](#)

beta\_ols, [3](#)  
beta\_to\_vector, [3](#)

conda\_binary(), [8](#)  
conesta, [4](#)  
conesta(), [11](#)  
cost, [5](#)

dist\_beta, [6](#)

fun\_lines, [6](#)

image\_sparse, [7](#)  
install\_pylearn\_parsimony, [8](#)

merge\_clusters, [9](#)  
mglasso, [9](#)  
mglasso(), [5](#)

plot\_clusterpath, [11](#)  
plot\_mglasso, [12](#)  
plot\_mglasso(), [11](#)  
precision\_to\_regression, [12](#)

reticulate::py\_install(), [8](#)

symmetrize, [13](#)