

# Package ‘CVglasso’

June 4, 2018

**Type** Package

**Title** Lasso Penalized Precision Matrix Estimation

**Version** 1.0

**Date** 2018-05-31

**Description** Estimates a lasso penalized precision matrix via the blockwise coordinate descent (BCD). This package is a simple wrapper around the popular 'glasso' package that extends and enhances its capabilities. These enhancements include built-in cross validation and visualizations.  
See Friedman et al (2008) <doi:10.1093/biostatistics/kxm045> for details regarding the estimation method.

**URL** <https://github.com/MGallow/CVglasso>

**BugReports** <https://github.com/MGallow/CVglasso/issues>

**License** GPL (>= 2)

**ByteCompile** TRUE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

**Imports** stats, parallel, foreach, ggplot2, dplyr, glasso

**Depends** doParallel

**Suggests** testthat

**NeedsCompilation** no

**Author** Matt Galloway [aut, cre]

**Maintainer** Matt Galloway <gall0441@umn.edu>

**Repository** CRAN

**Date/Publication** 2018-06-04 08:42:55 UTC

## R topics documented:

CVglasso . . . . .	2
plot.CVglasso . . . . .	5

CVglasso

*Penalized precision matrix estimation***Description**

Penalized precision matrix estimation using the graphical lasso (glasso) algorithm. Consider the case where  $X_1, \dots, X_n$  are iid  $N_p(\mu, \Sigma)$  and we are tasked with estimating the precision matrix, denoted  $\Omega \equiv \Sigma^{-1}$ . This function solves the following optimization problem:

**Objective:**  $\hat{\Omega}_\lambda = \arg \min_{\Omega \in S_+^p} \{Tr(S\Omega) - \log \det(\Omega) + \lambda \|\Omega\|_1\}$

where  $\lambda > 0$  and we define  $\|A\|_1 = \sum_{i,j} |A_{ij}|$ .

**Usage**

```
CVglasso(X = NULL, S = NULL, nlam = 10, lam.min.ratio = 0.01,
         lam = NULL, diagonal = FALSE, path = FALSE, tol = 1e-04,
         maxit = 10000, adjmaxit = NULL, K = 5, crit.cv = c("loglik", "AIC",
         "BIC"), start = c("warm", "cold"), cores = 1, trace = c("progress",
         "print", "none"), ...)
```

**Arguments**

<code>X</code>	option to provide a $n \times p$ data matrix. Each row corresponds to a single observation and each column contains $n$ observations of a single feature/variable.
<code>S</code>	option to provide a $p \times p$ sample covariance matrix (denominator $n$ ). If argument is <code>NULL</code> and <code>X</code> is provided instead then <code>S</code> will be computed automatically.
<code>nlam</code>	number of <code>lam</code> tuning parameters for penalty term generated from <code>lam.min.ratio</code> and <code>lam.max</code> (automatically generated). Defaults to 10.
<code>lam.min.ratio</code>	smallest <code>lam</code> value provided as a fraction of <code>lam.max</code> . The function will automatically generate <code>nlam</code> tuning parameters from <code>lam.min.ratio*lam.max</code> to <code>lam.max</code> in $\log_{10}$ scale. <code>lam.max</code> is calculated to be the smallest <code>lam</code> such that all off-diagonal entries in $\Omega$ are equal to zero ( $\alpha = 1$ ). Defaults to $1e-2$ .
<code>lam</code>	option to provide positive tuning parameters for penalty term. This will cause <code>nlam</code> and <code>lam.min.ratio</code> to be disregarded. If a vector of parameters is provided, they should be in increasing order. Defaults to <code>NULL</code> .
<code>diagonal</code>	option to penalize the diagonal elements of the estimated precision matrix ( $\Omega$ ). Defaults to <code>FALSE</code> .
<code>path</code>	option to return the regularization path. This option should be used with extreme care if the dimension is large. If set to <code>TRUE</code> , <code>cores</code> must be set to 1 and errors and optimal tuning parameters will be based on the full sample. Defaults to <code>FALSE</code> .
<code>tol</code>	convergence tolerance. Iterations will stop when the average absolute difference in parameter estimates is less than <code>tol</code> times multiple. Defaults to $1e-4$ .
<code>maxit</code>	maximum number of iterations. Defaults to $1e4$ .

<code>adjmaxit</code>	adjusted maximum number of iterations. During cross validation this option allows the user to adjust the maximum number of iterations after the first <code>lam</code> tuning parameter has converged. This option is intended to be paired with <code>warm</code> starts and allows for 'one-step' estimators. Defaults to <code>NULL</code> .
<code>K</code>	specify the number of folds for cross validation.
<code>crit.cv</code>	cross validation criterion ( <code>loglik</code> , <code>AIC</code> , or <code>BIC</code> ). Defaults to <code>loglik</code> .
<code>start</code>	specify <code>warm</code> or <code>cold</code> start for cross validation. Default is <code>warm</code> .
<code>cores</code>	option to run CV in parallel. Defaults to <code>cores = 1</code> .
<code>trace</code>	option to display progress of CV. Choose one of <code>progress</code> to print a progress bar, <code>print</code> to print completed tuning parameters, or <code>none</code> .
<code>...</code>	additional arguments to pass to <code>glasso</code> .

### Details

For details on the implementation of the 'glasso' function, see Tibshirani's website. <http://statweb.stanford.edu/~tibs/glasso/>.

### Value

returns class object `CVglasso` which includes:

<code>Call</code>	function call.
<code>Iterations</code>	number of iterations
<code>Tuning</code>	optimal tuning parameters ( <code>lam</code> and <code>alpha</code> ).
<code>Lambdas</code>	grid of lambda values for CV.
<code>maxit</code>	maximum number of iterations for outer (blockwise) loop.
<code>Omega</code>	estimated penalized precision matrix.
<code>Sigma</code>	estimated covariance matrix from the penalized precision matrix (inverse of <code>Omega</code> ).
<code>Path</code>	array containing the solution path. Solutions will be ordered by ascending lambda values.
<code>MIN.error</code>	minimum average cross validation error ( <code>cv.crit</code> ) for optimal parameters.
<code>AVG.error</code>	average cross validation error ( <code>cv.crit</code> ) across all folds.
<code>CV.error</code>	cross validation errors ( <code>cv.crit</code> ).

### Author(s)

Matt Galloway <[gall0441@umn.edu](mailto:gall0441@umn.edu)>

## References

- Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 'Sparse inverse covariance estimation with the graphical lasso.' *Biostatistics* 9.3 (2008): 432-441.
- Banerjee, Onureen, Ghauoui, Laurent El, and d'Aspremont, Alexandre. 2008. 'Model Selection through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data.' *Journal of Machine Learning Research* 9: 485-516.
- Tibshirani, Robert. 1996. 'Regression Shrinkage and Selection via the Lasso.' *Journal of the Royal Statistical Society. Series B (Methodological)*. JSTOR: 267-288.
- Meinshausen, Nicolai and Buhlmann, Peter. 2006. 'High-Dimensional Graphs and Variable Selection with the Lasso.' *The Annals of Statistics*. JSTOR: 1436-1462.
- Witten, Daniela M, Friedman, Jerome H, and Simon, Noah. 2011. 'New Insights and Faster computations for the Graphical Lasso.' *Journal of Computation and Graphical Statistics*. Taylor and Francis: 892-900.
- Tibshirani, Robert, Bien, Jacob, Friedman, Jerome, Hastie, Trevor, Simon, Noah, Jonathan, Taylor, and Tibshirani, Ryan J. 'Strong Rules for Discarding Predictors in Lasso-Type Problems.' *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*. Wiley Online Library 74 (2): 245-266.
- Ghaoui, Laurent El, Viallon, Vivian, and Rabbani, Tarek. 2010. 'Safe Feature Elimination for the Lasso and Sparse Supervised Learning Problems.' *arXiv preprint arXiv: 1009.4219*.
- Osborne, Michael R, Presnell, Brett, and Turlach, Berwin A. 'On the Lasso and its Dual.' *Journal of Computational and Graphical Statistics*. Taylor and Francis 9 (2): 319-337.
- Rothman, Adam. 2017. 'STAT 8931 notes on an algorithm to compute the Lasso-penalized Gaussssian likelihood precision matrix estimator.'

## See Also

[plot.CVglasso](#)

## Examples

```
# generate data from a sparse matrix
# first compute covariance matrix
S = matrix(0.7, nrow = 5, ncol = 5)
for (i in 1:5){
  for (j in 1:5){
    S[i, j] = S[i, j]^abs(i - j)
  }
}

# generate 100 x 5 matrix with rows drawn from iid N_p(0, S)
Z = matrix(rnorm(100*5), nrow = 100, ncol = 5)
out = eigen(S, symmetric = TRUE)
S.sqrt = out$vectors %*% diag(out$values^0.5)
S.sqrt = S.sqrt %*% t(out$vectors)
X = Z %*% S.sqrt

# lasso penalty CV
CVglasso(X)
```

---

plot.CVglasso	<i>Plot CVglasso object</i>
---------------	-----------------------------

---

### Description

Produces a plot for the cross validation errors, if available.

### Usage

```
## S3 method for class 'CVglasso'
plot(x, type = c("line", "heatmap"), footnote = TRUE,
     ...)
```

### Arguments

x	class object CVglasso
type	produce either 'heatmap' or 'line' graph
footnote	option to print footnote of optimal values. Defaults to TRUE.
...	additional arguments.

### Examples

```
# generate data from a sparse matrix
# first compute covariance matrix
S = matrix(0.7, nrow = 5, ncol = 5)
for (i in 1:5){
  for (j in 1:5){
    S[i, j] = S[i, j]^abs(i - j)
  }
}

# generate 100 x 5 matrix with rows drawn from iid N_p(0, S)
Z = matrix(rnorm(100*5), nrow = 100, ncol = 5)
out = eigen(S, symmetric = TRUE)
S.sqrt = out$vectors %*% diag(out$values^0.5)
S.sqrt = S.sqrt %*% t(out$vectors)
X = Z %*% S.sqrt

# produce line graph for CVglasso
plot(CVglasso(X))

# produce CV heat map for CVglasso
plot(CVglasso(X), type = 'heatmap')
```

# Index

CVglasso, [2](#)

plot.CVglasso, [4](#), [5](#)