Package ‘sparsenetgls’

May 30, 2024

Type Package

Title Using Gaussian graphical structure learning estimation in
generalized least squared regression for multivariate normal
regression

Version 1.22.0

Description The package provides methods of combining the graph structure learning and generalized
least squares regression to improve the regression estimation. The main function sparsenet-
gls() provides
solutions for multivariate regression with Gaussian distributed dependant variables and ex-
planatory variables
utilizing multiple well-
known graph structure learning approaches to estimating the precision matrix, and uses
a penalized variance covariance matrix with a distance tuning parameter of the graph struc-
ture in deriving the
sandwich estimators in generalized least squares (gls) regression. This package also pro-
vides functions for
assessing a Gaussian graphical model which uses the
penalized approach. It uses Receiver Operative Characteristics
curve as a visualization tool in the assessment.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0), Matrix, MASS

Imports methods, glmnet, huge, stats, graphics, utils

Suggests testthat, lme4, BiocStyle, knitr, rmarkdown, roxygen2 (>= 5.0.0)

NeedsCompilation no

RoxygenNote 6.0.1

biocViews ImmunoOncology,

GraphAndNetwork,Regression,Metabolomics,CopyNumberVariation,MassSpectrometry,Proteomics,Software,Visualization

bugReport https://github.com/superOmics/sparsenetgls/issues

VignetteBuilder knitr
The assess_direct() function

The assess_direct function is designed to evaluate the prediction accuracy of a Gaussian Graphical model (GGM) comparing with the true graph structure with a known precision matrix.

Usage

assess_direct(PREC_for_graph, OMEGA_for_graph, p)

Arguments

PREC_for_graph  It is the known precision matrix which is used to assess the estimated precision matrix from GGM.
OMEGA_for_graph  It is the estimated precision matrix from a GGM.
p  It is an integer representing the number of dimension of both the known and estimated precision matrix.
Value

Return the list of assessment results including sensitivity, specificity, NPV(test negative), PPV(test positive), true positive and true negative.

Examples

```r
prec1 <- matrix(c(0,2,3,1,0,0.5,0,0,0.4),nrow=3,ncol=3)
prec0 <- matrix(c(0,1,2,1,0.5,0.2,0,1,1),nrow=3,ncol=3)
assessresult <- assess_direct(prec1,prec0,p=3)
```

**Description**

`bandprec` and `bandvar` store the precision matrix and variance covariance matrix with the band diagonal structure.

**Usage**

`data("bandprec")`

**Format**

A data frame with 50 observations on the following 50 variables.

<table>
<thead>
<tr>
<th>V1</th>
<th>a numeric vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>V2</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V3</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V4</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V5</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V6</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V7</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V8</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V9</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V10</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V11</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V12</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V13</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V14</td>
<td>a numeric vector</td>
</tr>
<tr>
<td>V15</td>
<td>a numeric vector</td>
</tr>
</tbody>
</table>
data(bandprec)
## maybe str(bandprec) ; plot(bandprec) ...
**convertbeta**  
*The convertbeta() function*

**Description**  
The `convertbeta` function is designed to convert the regression coefficients derived from the standardized data.

**Usage**  
`convertbeta(X, Y, q, beta0)`

**Arguments**
- **X**: It is a dataset of explanatory variables.
- **Y**: It is the multivariate response variables.
- **q**: It is an integer representing the number of explanatory variables and intercept.
- **beta0**: The vector contains the regression coefficients result from `sparsenetgls`.

**Value**  
Return the list of converted regression coefficients of the explanatory variables 'betaconv' and intercept value 'betaconv_int'.

**Examples**
```r
X <- mvrnorm(n=20,mu=rep(0,5),Sigma=Diagonal(5,rep(1,5)))
Y <- mvrnorm(n=20,mu=rep(0.5,10),Sigma=Diagonal(10,rep(1,10)))
fitmodel <- sparsenetgls(responsedata=Y,predictdata=X,nlambda=5,ndist=2,
method='elastic')
#Example of converting the regression coef of the first lambda
convertbeta(X=X,Y=Y,q=5+1,beta0=fitmodel$beta[,1])
```

---

**glassonet2**  
*The glassonet2() function*

**Description**

The `glassonet2` function is designed to learn the graph structure, the corresponding precision matrix and covariance matrix by using the graph lasso method.

**Usage**
```
glassonet2(Y0, nlambda = nlambda, lambda.min.ratio = 0.001, method)
```
**lassoglmnet**

**Arguments**

- **Y0**
  The data matrix for the GGM model.

- **nlambda**
  The number of interval used in the penalized path in lasso and elastics. It results in the number of lambda values to be used in the penalization. The default value is nlambda assigned in the parent function sparsenetgls().

- **lambda.min.ratio**
  It is the default parameter set in function huge() in the package 'huge'. Quoted from huge(), it is the minimal value of lambda, being a fraction of the upper bound (MAX) of the regularization/thresholding parameter that makes all the estimates equal to 0. The default value is 0.001.

- **method**
  There are two options for the method parameter which is provided in the huge() function. One is 'glasso' and the other one is 'mb'.

**Value**

Return the precision matrix 'OMEGAMATRIX', penalized path parameter lambda 'lambda' and covariance matrix 'COVMATRIX'.

**Examples**

```r
n=20
VARknown <- rWishart(1, df=4, Sigma=matrix(c(1,0,0,0,1,0,0,0,1), nrow=3,ncol=3))
Y0 <- mvrnorm(n=n,mu=rep(0.5,3),Sigma=VARknown[,1])
fitglasso <- glassonet2(Y0=Y0,nlambda=5,method='glasso')
```

---

**lassoglmnet**

*The lassoglmnet() function*

**Description**

The lassoglmnet function is designed to learn the graph structure by using the lasso and elastics net methods.

**Usage**

lassoglmnet(Y0, nlambda = 10, alpha)

**Arguments**

- **Y0**
  The data matrix for the GGM model.

- **nlambda**
  The number of interval used in the penalized path in lasso and elastics. It results in the number of lambda values to be used in the penalization. The default value is 10.
alpha

The value to be used in enet, it has values between 0 and 1. The value of 0 is corresponding to l-1 penalization, and 1 is corresponding to the l-2 regularization (Ridge regression). The other values between 0 and 1 will result in a combination of l1-l2 norm regularization named as elastic net.

Value

Return the regression coefficients of glmnet 'coef_glmnet', residuals from the glmnet 'resid_glmnet' and lambda.

Examples

n=20
VARknown <- rWishart(1,df=4,Sigma=matrix(c(1,0,0,0,1,0,0,0,1),nrow=3,ncol=3))
Y0 <- mvrnorm(n=n,mu=rep(0.5,3),Sigma=VARknown[,1])
fitlasso <- lassoglmmnet(Y0=Y0,alpha=0.5)

---

path_result_for_roc  The path_result_for_roc() function

Description

The path_result_for_roc function is designed to evaluate the prediction accuracy of a series Gaussian Graphical models (GGM) comparing to the true graph structure. The GGM must use a l-p norm regularizations (p=1,2) with the series of solutions conditional on the regularization parameter.

Usage

path_result_for_roc(PREC_for_graph, OMEGA_path, pathnumber)

Arguments

PREC_for_graph  It is the known precision matrix which is used to assess the estimated precision matrix from GGM.
OMEGA_path  It is a matrix comprising of a series estimated precision matrices from a GGM model using a penalized path based on a range of structure parameters (i.e. \( \lambda, \in [0,1] \)).
pathnumber  It represents the number of graph models (i.e. \( \lambda \)) for the evaluation. The value of pathnumber can be the same number used in a penalized path.

Value

Return the list of assessment results for a series of precision matrices. The results include sensitivity/specificity/NPV/PPV
Examples

```r
prec1 <- matrix(c(0,2,3,1,0,0.5,0,0,0.4),nrow=3,ncol=3)
Omega_est <- array(dim=c(3,3,3))
Omega_est[,,1] <- matrix(c(0,1,2,1,0.5,0.2,0,1,1),nrow=3,ncol=3)
Omega_est[,,2] <- matrix(c(0,1,0,1,0.5,0.2,0,1,1),nrow=3,ncol=3)
Omega_est[,,3] <- matrix(c(0,1,0,1,0,0.2,0,1,1),nrow=3,ncol=3)
rocpath <- path_result_for_roc(PREC_for_graph=prec1,OMEGA_path=Omega_est,
pathnumber=3)
```

---

### plotsngls

**The plotsngls() function**

**Description**

The plotsngls function is designed to provide the line plots of variance of regression coefficients vs. values of penalized parameter lambda in gls regression, when the tuning parameter d is the maximal value. It also provides the graph structure of the estimated precision matrix in the penalized path.

**Usage**

```r
plotsngls(
  fitgls,
  lineplot = FALSE,
  nrow,
  ncol,
  structplot = TRUE,
  ith_lambda = 1
)
```

**Arguments**

- `fitgls`  
  It is a returning object of the sparsnetgls() multivariate generalized least squared regression function.
- `lineplot`  
  It is a logical indicator. When value=TRUE, it will provide line plot.
- `nrow`  
  It is a graph parameter representing number of rows in the lineplot.
- `ncol`  
  It is a graph parameter representing number of columns in the lineplot.
- `structplot`  
  It is a logical indicator. When value=TRUE, it will provide the structure plot of the specified precision matrix from the series of the sparsenetgls results.
- `ith_lambda`  
  It is the number for the specified precision matrix to be used in the structplot. It represents the ordering number in the precision matrix series from sparsenetgls.

**Value**

Return a plot subject for sparsnetgls including the plot of variance vs lambda and graph structure of the precision matrix estimates.
Examples

```r
ndox=5;p=3;n=200
VARknown <- rWishart(1, df=4, Sigma=matrix(c(1,0,0,0,1,0,0,0,1),
nrow=3,ncol=3))
normc <- mvrnorm(n=n,mu=rep(0,p),Sigma=VARknown[,1])
Y0=normc
#u-beta
u <- rep(1,ndox)
X <- mvrnorm(n=n,mu=rep(0,ndox),Sigma=Diagonal(ndox,rep(1,ndox)))
X00 <- scale(X,center=TRUE, scale=TRUE)
X0 <- cbind(rep(1,n),X00)
#Add predictors of simulated CNA
abundance1 <- scale(Y0,center=TRUE, scale=TRUE)+as.vector(X00%*%as.matrix(u))

#sparsenetgls()
fitgls <- sparsenetgls(responsedata=abundance1,predictdata=X00,
nlambda=5,ndist=4,method='lasso')
plotsngls(fitgls, ith_lambda=5)
#plotsngls(fitgls,lineplot=TRUE,structplot=FALSE,nrow=2,ncol=3)
```

---

### plot_roc

**The plot_roc() function**

**Description**

The `plot_roc` function is designed to produce the Receiver Operative Characteristics (ROC) Curve for visualizing the prediction accuracy of a Gaussian Graphical model (GGM) to the true graph structure. The GGM must use a l-p norm regularizations (p=1,2) with the series of solutions conditional on the regularization parameter.

**Usage**

```r
plot_roc(result_assessment, group = TRUE, ngroup = 0, est_names)
```

**Arguments**

- `result_assessment`:
  It is the list result from function `path_result_for_roc()` which has five-dimensions recording the path number (i.e. the order of λ), the sensitivity, the specificity, the Negative predicted value (NPV) and the Positive predicted value (PPV) respectively.

- `group`:
  It is a logical parameter indicating if the `result_assessment` is for several GGM models. When it is TRUE, it produceS the ROC from several GGM models. when it is FALSE, it only produces a ROC for one model.

- `ngroup`:
  It is an integer recording the number of models when group is TRUE.

- `est_names`:
  It is used for labeling the GGM model in legend of ROC curve.


**Value**

Return the plot of Receiver Operational Curve

**Examples**

```r
test_example <- matrix(c(0,2,3,1,0,0.5,0,0,0.4), nrow=3, ncol=3)
Omega_est <- array(dim=c(3,3,3))
Omega_est[,,1] <- matrix(c(1,1,1,0.2,0.5,0.2,2,0.2,0.3), nrow=3, ncol=3)
Omega_est[,,2] <- matrix(c(1,1,1,0,0,0,1,0,0), nrow=3, ncol=3)
Omega_est[,,3] <- matrix(c(0,0,0,0,0,0,0,0,0), nrow=3, ncol=3)
roc_path_result <- path_result_for_roc(PREC_for_graph=test_example, OMEGA_path=Omega_est, pathnumber=3)
plot_roc(result_assessment=roc_path_result, group=FALSE, ngroup=0, est_names='test example')
```

---

**Description**

The sparsenetgls function is a combination of the graph structure learning and generalized least square regression. It is designed for multivariate regression uses penalized and/or regularised approach to deriving the precision and covariance Matrix of the multivariate Gaussian distributed responses. Gaussian Graphical model is used to learn the structure of the graph and construct the precision and covariance matrix. Generalized least squared regression is used to derive the sandwich estimation of variances-covariance matrix for the regression coefficients of the explanatory variables, conditional on the solutions of the precision and covariance matrix.

**Usage**

```r
sparsenetgls(
  respondedata,
  predictdata,
  nlambda = 10,
  ndist = 5,
  method = c("lasso", "glasso", "elastic", "mb"),
  lambda.min.ratio = 1e-05
)
```

**Arguments**

- **respondedata**
  It is a data matrix of multivariate normal distributed response variables. Each row represents one observation sample and each column represents one variable.

- **predictdata**
  It is a data matrix of explanatory variables and has the same number of rows as the response data.
nlambda

It is an interger recording the number of lambda value used in the penalized path for estimating the precision matrix. The default value is 10.

ndist

It is an interger recording the number of distant value used in the penalized path for estimating the covariance matrix. The default value is 5.

method

It is the option parameter for selecting the penalized method to derive the precision matrix in the calculation of the sandwich estimator of regression coefficients and their variance-covariance matrix. The options are 'glasso', 'lasso', 'elastic', and 'mb'. 'glasso' use the graphical lasso method documented in Yuan and lin (2007) and Friedman, Hastie et al (2007). It used the imported function from R package 'huge'. 'lasso' use the penalized liners regression among the response variables (Y[,j]~Y[,j]+...Y[,j-1],Y[,j+1]+...Y[,p]) to estimate the precision matrix. 'elastic' uses the enet-regularized linear regression among the response variables to estimate the precision matrix. Both of these methods utilize the coordinate descending algorithm documented in Friedman, J., Hastie, T. and Tibshirani, R. (2008) and use the imported function from R package 'glmnet'. 'mb' use the Meinshausen and Buhlmann penalized linear regression and the neighbourhood selection with the lasso approach (2006) to select the covariance terms and derive the corresponding precision matrix ; It uses the imported function from 'huge' in function sparsenetgls().

lambda.min.ratio

It is the default parameter set in function huge() in the package 'huge'. Quoted from huge(), it is the minial value of lambda, being a fraction of the uppperbound (MAX) of the regularization/thresholding parameter that makes all the estimates equal to 0. The default value is 0.001. It is only applicable when 'glasso' and 'mb' method is used.

Value

Return the list of regression results including the regression coefficients, array of variance-covariance matrix for different lambda and distance values, lambda and distance (power) values, bic and aic for model fitting, and the list of precision matrices on the penalized path.

Examples

nndo=5; p=3; n=1000
VARknown <- rWishart(1, df=4, Sigma=matrix(c(1,0,0,0,1,0,0,0,1), nrow=3,ncol=3))
normc <- mvrnorm(n=n,mu=rep(0,p),Sigma=VARknown[,,1])
Y0=normc
#u-beta
u <- rep(1,nndo)
X <- mvrnorm(n=n,mu=rep(0,nndo),Sigma=Diagonal(nndo,rep(1,nndo)))
X00 <- scale(X,center=TRUE,scale=TRUE)
X0 <- cbind(rep(1,n),X00)
#Add predictors of simulated CNA
abundance1 <- scale(Y0,center=TRUE,scale=TRUE)+as.vector(X00%*%as.matrix(u))

#sparsenetgls()
fitgls <- sparsenetgls(responsedata=abundance1,predictdata=X00,
nlambda=5, ndist=2, method='elastic')
nlambda=5

# rescale regression coefficients from sparsenetgls
#betagls <- matrix(nrow=nlambda, ncol=ndox+1)
#for (i in seq_len(nlambda))
#betagls[i,] <- convertbeta(Y=abundance1, X=X00, q=ndox+1,
#beta0=fitgls$beta[,i])$betaconv
Index

* datasets
  bandprec, 3
  assess_direct, 2
  bandprec, 3
  convertbeta, 5
  glassonet2, 5
  lassoglmnet, 6
  path_result_for_roc, 7
  plot_roc, 9
  plotsngls, 8
  sparsenetgls, 10