Package ‘sparseBC’

October 14, 2022

Type Package

Title Sparse Biclustering of Transposable Data

Version 1.2

Date 2019-03-18

Author Kean Ming Tan

Maintainer Kean Ming Tan <keanming@u.washington.edu>

Depends glasso

Imports fields

Description Implements the sparse biclustering proposal of Tan and Witten (2014), Sparse biclustering of transposable data. Journal of Computational and Graphical Statistics 23(4):985-1008.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2019-03-19 05:40:05 UTC

R topics documented:

sparseBC-package *Fit sparse biclustering and matrix-variate normal biclustering*

Description

This package is called sparseBC, for "Sparse biclustering". It implements two methods:Sparse biclustering and matrix-variate normal biclustering. All are described in the paper "Sparse biclustering of tranposable data", by KM Tan and D Witten (2014), *Journal of Computational and Graphical Statistics*.

The main functions are as follows: (1) sparseBC (2) matrixBC

The first function, sparseBC, performs sparse biclustering. matrixBC performs matrix-variate normal biclustering. There are also cross-validation functions for tuning parameter that controls the sparsity level of the estimated mean matrix: sparseBC.BIC and matrixBC.BIC. Function that choose the number of biclusters K and R are also included for sparseBC, called sparseBC.choosekr.

Details

Package:	sparseBC
Type:	Package
Version:	1.1
Date:	2015-02-09
License:	GPL (>=2.0)
LazyLoad:	yes

The package includes the following functions:

`sparseBC`:	Perform sparse biclustering
`sparseBC.choosekr`:	Cross-validation to select the number of row and column clusters
`sparseBC.BIC`:	Select sparsity tuning parameter for sparseBC
`summary.sparseBC`:	Display information for the object sparseBC
`image.sparseBC`:	Image plot for the estimated bicluster mean matrix
`matrixBC`:	Perform matrix-variate normal biclustering
`matrixBC.BIC`:	Select sparsity tuning parameter for matrixBC

Author(s)

Kean Ming Tan

Maintainer: Kean Ming Tan <keanming@u.washington.edu>

References

KM Tan and D Witten (2014) Sparse biclustering of transposable data. *Journal of Computational and Graphical Statistics* 23(4):985-1008.

See Also

`sparseBC` `matrixBC`

Examples

```
# An example that violates the assumption of contiguous biclusters
# Create mean matrix and the data matrix
#set.seed(5)
#u<-c(10,9,8,7,6,5,4,3,rep(2,17),rep(0,75))
#v<-c(10,-10,8,-8,5,-5,rep(3,5),rep(-3,5),rep(0,34))
#u<-u/sqrt(sum(u^2))
#v<-v/sqrt(sum(v^2))
#d<-50
#mus<-d*tcrossprod(u,v)
#binaryX<-(mus!=0)*1
#X<-mus+matrix(rnorm(100*50),100,50)
#X<-X-mean(X)

# The number of biclusters are chosen automatically
# Commented out for short run-time
#KR<-sparseBC.choosekr(X,1:6,1:6,0,0.1,trace=TRUE)
#k<-KR$estimated_kr[1]
#r<-KR$estimated_kr[2]

# The value of lambda is chosen automatically
#lambda<-sparseBC.BIC(X,k,r,c(0,10,20,30,40,50))$lambda

# Perform sparse biclustering using the K, R, and lambda chosen
#biclustering<-sparseBC(X,k,r,lambda)

# Display some information on the object sparseBC
#summary(biclustering)

# Plot the estimated mean matrix from sparseBC
#image(biclustering)
```

`image.sparseBC` *Image plot of an object of class* `sparseBC` *or* `matrixBC`

Description

This function plots a `sparseBC` or `matrixBC` object — the estimated bicluster mean matrix for the observations and features

Usage

```
## S3 method for class 'sparseBC'
image(x, labelx = TRUE, labely = TRUE, arrangex = FALSE, arrangey = FALSE, ...)
```

Arguments

`x`	an object of class `sparseBC` or `matrixBC`
`labelx`	label the x-axis with the features number. The default is labelx=TRUE
`labely`	label the y-axis with the observations number. The default is labely=TRUE
`arrangex`	Rearrange the features/columns such that they are grouped into their respective clusters for the image plot. The default is arrangex=FALSE
`arrangey`	Rearrange the observations/rows such that they are grouped into their respective clusters for the image plot. The default is arrangey=FALSE.
`...`	additional parameters to be passed to `image`

Details

This function plots the estimated mean matrix from `sparseBC` or `matrixBC`. The columns are the features and the rows are the observations.

Author(s)

Kean Ming Tan and Daniela Witten

References

#

See Also

`sparseBC` `matrixBC` `summary.sparseBC` `sparseBC.BIC` `sparseBC.choosekr`

Examples

```
# See example in sparseBC
```

`lung` *Lung cancer gene expression data*

Description

This data set consists of gene expression values of a subset of 5000 genes with the highest variance of the 12 625 genes measured using the Affymetrix 95av2 GeneChip on a set of 56 samples - 20 pulmonary carcinoid samples (Carcinoid), 6 small cell lung carcinoma samples (SmallCell), 13 colon cancer metastasis samples (Colon), and 16 normal lung samples (Normal). This example is used in Lee, Shen, Huang, and Marron (2010).

Usage

```
data(lung)
```

Format

The format is an n x p matrix: (1) the rows are the samples - rownames containing the class labels. (2) the columns are the genes - colnames containing the affymetrix gene ids.

Source

This data set was published in the following paper:

Bhattacharjee, A., Richards, W., Staunton, J., Li, C., Monti, S., Vasa, P., Ladd, C., Beheshti, J., Bueno, R., Gillette, M., and others. (2001) Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. Proceedings of the National Academy of Sciences of the United States of America, 98(24), 13790–13795.

It is publicly available at http://www.pnas.org/content/98/24/13790/suppl/DC1

References

Bhattacharjee et al. (2001) Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. Proceedings of the National Academy of Sciences of the United States of America, 98(24), 13790–13795

Used as an example in Lee, Shen, Huang, and Marron (2010), 'Biclustering via sparse singular value decomposition', Biometrics 66(4), 1087–1095.

Used as an example in KM Tan and D Witten (2014) Sparse biclustering of transposable data. *Journal of Computational and Graphical Statistics* 23(4):985-1008.

Examples

```
data(lung)
```

`matrixBC` *MVN biclustering*

Description

This function performs MVN biclustering on a n by p matrix. Details are given in Tan and Witten (2014).

Usage

```
matrixBC(x, k, r, lambda, alpha, beta, nstart = 20, Cs.init = NULL,
 Ds.init = NULL, max.iter = 50, threshold = 1e-04, Sigma.init = NULL,
 Delta.init = NULL, center=TRUE)
```

Arguments

`x`	Data matrix; samples are rows and columns are features. Cannot contain missing values.
`k`	The number of row clusters, i.e., the number of clusters for the observations.
`r`	The number of column clusters, i.e., the number of clusters for the features.
`lambda`	Non-negative regularization parameter for lasso on the mean of each bicluster. lambda=0 means no regularization.
`alpha`	Non-negative regularization parameter for the graphical lasso to estimate the covariance matrix of the samples. alpha=0 means no regularization. alpha>0 is recommended.
`beta`	Non-negative regularization parameter for the graphical lasso to estimate the covariance matrix of the features. beta=0 means no regularization. beta>0 is recommended.
`nstart`	The number of random initialization sets used in the kmeans function. The default is 20.
`Cs.init`	Starting values for the row labels. The default value is NULL – kmeans clustering is performed to estimate the row labels.
`Ds.init`	Starting values for the column labels. The default value is NULL – kmeans clustering is performed to estimate the column labels.
`max.iter`	Maximum number of iterations. The default value is 50 iterations.
`threshold`	Threshold value for convergence. The default is 1e-4.
`Sigma.init`	Starting values for the covariance matrix of the observations. The default value is NULL – the graphical lasso as described in Friedman, Hastie, and Tibshirani (2008) is performed to estimate the covariance matrix of the observations.
`Delta.init`	Starting values for the covariance matrix of the features. The default value is NULL – the graphical lasso as described in Friedman, Hastie, and Tibshirani (2008) is performed to estimate the covariance matrix of the features.
`center`	Mean center the data matrix before performing sparse biclustering. The default is TRUE.

Details

This implements MVN biclustering using Algorithm (3) described in Tan and Witten (2014) 'Sparse biclustering of transposable data'. This approach takes into account the correlation among the features within the same cluster and also takes into account the correlation among the observations within the same cluster. The row labels for the observations and column labels for the features are estimated and the mean of each bicluster is encouraged to be sparse using the lasso penalty. Details are given in Algorithm (3) in Tan and Witten (2014).

If Sigma.init and Delta.init are NULL, the graphical lasso in Friedman, Hastie, and Tibshirani (2008) is used to estimated the covariance matrix of the observations and features. If Sigma.init is provided, then the covariance matrices would not be updated in the algorithm. Note that when Sigma and Delta equal the identity matrix up to a scaling factor, this approach is exactly that of sparse biclustering and the function sparseBC should be used.

Note that most of the computation time comes from the graphical lasso algorithm. We recommend setting the tuning parameters alpha and beta to be large so that the graphical lasso can be implemented efficiently (see the glasso package). When n > p, alpha=0 will return an error. Similarly, when p > n, beta=0 will return an error.

If center=TRUE, the data matrix x is mean centered before performing sparse biclustering. The reported mean matrix mus is the addition of the substracted mean, mean(x), and the estimated mean matrix from sparse biclustering on the mean centered data.

Value

an object of class matrixBC.

Among some internal variables, this object includes the elements

`Cs`	Cs is the output for the row labels.
`Ds`	Ds is the output for the column labels.
`mus`	mus is the estimated mean matrix for the entire matrix.
`Mus`	Mus is the estimated mean matrix for each bicluster.
`Sigma`	Sigma is the estimated covariance matrix of the observations.
`Delta`	Delta is the estimated covariance matrix of the features.
`objs`	objs is the maximized objective value of the negative l1 penalized log-likelihood of the matrix-variate normal distribution.
`iteration`	The number of iterations until convergence.

Author(s)

Kean Ming Tan and Daniela Witten

References

KM Tan and D Witten (2014) Sparse biclustering of transposable data. *Journal of Computational and Graphical Statistics* 23(4):985-1008.

J Friedman, T Hastie, and R Tibshirani (2008). Sparse inverse covariance estimation with the lasso. *Biostatistics* 9, 432–441.

See Also

`sparseBC` `matrixBC.BIC` `summary.matrixBC` `image.matrixBC`

Examples

```
# Lung cancer data
# Not run to save time
#data(lung)
#truecluster<-as.numeric(as.factor(rownames(lung)))
#cancersd<-apply(lung,2,sd)
# Pick the top 400 genes that have the largest standard deviation
#lung<-lung[,rank(cancersd)>=length(cancersd)-399]
```

```
# Example of MVN Biclustering
#set.seed(5)
#res<-matrixBC(lung,k=4,r=10,lambda=60,alpha=0.4,beta=0.4)
# one misclassification
#res$Cs

# lambda chosen such that the estimated mean matirx ofsparseBC has a
# similar number of nonzero as matrixBC
#res2<-sparseBC(lung,k=4,r=10,lambda=230)
# a few observations are being misclassified
#res2$Cs

# print information from the object matrixBC
#summary(res)

# Plot the estimated mean matris for the object matrixBC
#image(res)
```

matrixBC.BIC — *Do tuning parameter (lambda) selection for MVN biclustering via BIC criterion*

Description

We assume that both K and R are known. A range of values of lambda is usually considered - value that results in the lowest BIC is selected.

Usage

```
matrixBC.BIC(x, k, r, lambda, alpha = 0.2, beta = 0.2, nstart = 20,
Sigma.init = NULL, Delta.init = NULL)
```

Arguments

`x`	Data matrix; samples are rows and columns are features. Cannot contain missing values.
`k`	The number of row clusters, i.e., the number of clusters for the observations.
`r`	The number of column clusters, i.e., the number of clusters for the features.
`lambda`	A range of values of tuning parameters to be considered. All values must be non-negative.
`alpha`	Non-negative regularization parameter for the graphical lasso to estimate the covariance matrix of the samples. lambda=0 means no regularization.
`beta`	Non-negative regularization parameter for the graphical lasso to estimate the covariance matrix of the features. lambda=0 means no regularization.

`nstart`	The number of random initialization sets used in the kmeans function. The default is 20.
`Sigma.init`	Starting values for the covariance matrix of the observations. The default value is NULL – the graphical lasso as described in Friedman, Hastie, and Tibshirani (2007) is performed to estimate the covariance matrix of the observations.
`Delta.init`	Starting values for the covariance matrix of the features. The default value is NULL – the graphical lasso as described in Friedman, Hastie, and Tibshirani (2007) is performed to estimate the covariance matrix of the features.

Details

This implements the tuning parameter selection for MVN biclustering using BIC criterion as described in Section 5.2 in Tan and Witten (2014) 'Sparse biclustering of transposable data'. The BIC criterion is BIC = np x log(RSS) + np log(q) where RSS is the usual residual sum of squares, and q is the number of non-zero bicluster mean in the output of matrixBC. We select the value of lambda that leads to the smallest BIC.

The data is centered to have mean 0 in this function.

Value

`lambda`	Value of lambda that results in lowest BIC.
`BIC`	BIC values for a range of tuning parameters considered.
`nonzeromus`	Number of nonzero bicluster means for a range of tuning parameters considered.

Author(s)

Kean Ming Tan and Daniela Witten

References

KM Tan and D Witten (2014) Sparse biclustering of transposable data. *Journal of Computational and Graphical Statistics* 23(4):985-1008.

See Also

`matrixBC`

Examples

```
########### Create data matrix
#x <- matrix(rnorm(20*30),nrow=20,ncol=30)

########### Perform sparseBC.BIC to select lambda
#lambda<-matrixBC.BIC(x,k=2,r=2,lambda=c(0,10,20,30,40),alpha=0.2,beta=0.2)$lambda
```

`sparseBC` *Sparse biclustering*

Description

This function performs sparse biclustering on an n by p matrix. Details are given in Tan and Witten (2014).

Usage

```
sparseBC(x, k, r, lambda, nstart = 20, Cs.init = NULL, Ds.init = NULL,
 max.iter = 1000,threshold=1e-10,center=TRUE)
```

Arguments

`x`	Data matrix; samples are rows and columns are features. Cannot contain missing values.
`k`	The number of row clusters, i.e., the number of clusters for the observations.
`r`	The number of column clusters, i.e., the number of clusters for the features.
`lambda`	Non-negative regularization parameter for lasso on the mean of each bicluster. lambda=0 means no regularization.
`nstart`	The number of random initialization sets used in the kmeans function. The default is 20.
`Cs.init`	Starting values for the row labels. The default value is NULL – kmeans clustering is performed to estimate the row labels.
`Ds.init`	Starting values for the column labels. The default value is NULL – kmeans clustering is performed to estimate the column labels.
`max.iter`	Maximum number of iterations. The default value is 1000 iterations.
`threshold`	Threshold value for convergence. The default is 1e-10.
`center`	Mean center the data matrix before performing sparse biclustering. The default is TRUE.

Details

This implements sparse biclustering using Algorithm (1) described in Tan and Witten (2014) 'Sparse biclustering of transposable data', which estimates the row labels for the observations and column labels for the features. The mean of each bicluster is encouraged to be sparse using the lasso penalty. Details are given in Algorithm (1) in Tan and Witten (2014).

If center=TRUE, the data matrix x is mean centered before performing sparse biclustering. The reported mean matrix mus is the addition of the substracted mean, mean(x), and the estimated mean matrix from sparse biclustering on the mean centered data.

Note that center=TRUE will not give any estimated mean to be zero, unless the data is initially centered to have mean(x)=0. Instead, when center=TRUE, elements of the mean matrix are shrunken towards mean(x).

Value

an object of class sparseBC.

Among some internal variables, this object includes the elements

`Cs`	Cs is the output for the row labels.
`Ds`	Ds is the output for the column labels.
`objs`	objs is the minimized objective value of the l1 penalized log-likelihood.
`mus`	mus is the estimated mean matrix for the entire matrix.
`Mus`	Mus is the estimated mean matrix for each bicluster.
`iteration`	The number of iterations until convergence.

Author(s)

Kean Ming Tan and Daniela Witten

References

KM Tan and D Witten (2014) Sparse biclustering of transposable data. *Journal of Computational and Graphical Statistics* 23(4):985-1008.

See Also

`sparseBC.BIC` `sparseBC.choosekr` `summary.sparseBC` `image.sparseBC`

Examples

```
###############################################
# Example from Figure 1 in the manuscript
# A toy example to illustrate the results from k-means and sparse biclustering
###############################################

# Generate the data matrix x
set.seed(1)
n<-100
p<-200
k<-5
r<-5
truthCs<-rep(1:k, each=(n/k))
truthDs<-rep(1:r, each=(p/r))
mus<-runif(k*r,-3,3)
mus<-matrix(c(mus),nrow=k,ncol=r,byrow=FALSE)
x<-matrix(rnorm(n*p,mean=0,sd=5),nrow=n,ncol=p)

# Generate the mean matrix
musmatrix<-matrix(NA,nrow=n,ncol=p)
for(i in 1:max(truthCs)){
  for(j in 1:max(truthDs)){
  x[truthCs==i,truthDs==j]<-x[truthCs==i,truthDs==j]+mus[i,j]
  musmatrix[truthCs==i,truthDs==j]<-mus[i,j]
  }
}

# Perform kmeans on the row and columns and calculate its mean
km.Cs<-kmeans(x,k,nstart=20)$cluster
km.Ds<-kmeans(t(x),r,nstart=20)$cluster
km.mus<-matrix(NA,nrow=n,ncol=p)
for(i in 1:n){
  for(j in 1:p){
  km.mus[i,j]<-mean(x[km.Cs==km.Cs[i],km.Ds==km.Ds[j]])
  }
}

# Perform sparse biclustering with 5 row clusters and 5 column clusters and lambda=0
bicluster<-sparseBC(x,5,5,0)


# Display some information on the object sparseBC
summary(bicluster)


# Image plots to illustrate the estimated mean matrix
par(mfrow=c(2,2))
image(t(x),main="x")
image(t(musmatrix),main="Mean Matrix")
image(t(km.mus),main="Kmeans")
image(t(bicluster$mus),main="sparseBC")

# Built-in image plot for object sparseBC
image(bicluster)
```

`sparseBC.BIC` *Do tuning parameter (lambda) selection for sparse biclustering via BIC criterion*

Description

We assume that both K and R are known. A range of values of lambda is usually considered - value that results in the lowest BIC is selected.

Usage

```
sparseBC.BIC(x, k, r, lambda)
```

Arguments

x	Data matrix; samples are rows and columns are features. Cannot contain missing values.
k	The number of row clusters, i.e., the number of clusters for the observations.

`r`	The number of column clusters, i.e., the number of clusters for the features.
`lambda`	A range of values of tuning parameters to be considered. All values must be non-negative.

Details

This implements the tuning parameter selection using BIC criterion for sparse biclustering as described in Section 5.2 in Tan and Witten (2014) 'Sparse biclustering of transposable data'. The BIC criterion is BIC = np x log(RSS) + np log(q) where RSS is the usual residual sum of squares, and q is the number of non-zero bicluster mean in the output of sparseBC. We select the value of lambda that leads to the smallest BIC.

The data is centered to have mean 0 in this function.

Value

`lambda`	Value of lambda that results in lowest BIC.
`BIC`	BIC values for a range of tuning parameters considered.
`nonzeromus`	Number of nonzero bicluster means for a range of tuning parameters considered.

Author(s)

Kean Ming Tan and Daniela Witten

References

KM Tan and D Witten (2014) Sparse biclustering of transposable data. *Journal of Computational and Graphical Statistics* 23(4):985-1008.

See Also

`sparseBC` `sparseBC.choosekr`

Examples

```
########### Create data matrix
x <- matrix(rnorm(20*30),nrow=20,ncol=30)

########### Perform sparseBC.BIC to select lambda
lambda<-sparseBC.BIC(x,k=2,r=2,lambda=c(0,10,20,30,40))$lambda
```

sparseBC.choosekr *Do tuning parameter K and R selection for sparse biclustering via cross-validation.*

Description

Perform cross-validation to select K (number of row clusters) and R (number of column clusters) for sparse biclustering. We assume that lambda is known.

Usage

```
sparseBC.choosekr(x, k, r, lambda, percent = 0.1, trace=FALSE)
```

Arguments

`x`	Data matrix; samples are rows and columns are features. Cannot contain missing values.
`k`	A range of values of K to be considered. Values considered must be an increasing sequence.
`r`	A range of values of R to be considered. Values considered must be an increasing sequence.
`lambda`	Non-negative regularization parameter for lasso. lambda=0 means no regularization.
`percent`	Percentage of elements of x to be left out for cross-validation. 1 must be divisible by the specified percentage. The default value is 0.1.
`trace`	Print out progress as iterations are performed. Default is FALSE.

Details

The function performs cross-validation as described in Algorithm (2) in Tan and Witten (2014) 'Sparse biclustering of transposable data'. Briefly, it works as follows: (1) some percent of the elements of x is removed at random from the data matrix - call those elements missing elements, (2) the missing elements are imputed using the mean of the other elements of the matrix, (3) sparse biclustering is performed with various values of K and R and the mean matrix is estimated, (4) calculate the sum of squared error of the missing values between x and the estimated mean matrix. This procedure is repeated 1/percent times. Finally, we select K and R based on the criterion described in Algorithm (2) in Tan and Witten (2014). A similar procedure is used in Witten et al (2009) 'A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis'.

Note that sparseBC is run with center=TRUE in this function.

Value

`estimated_kr`	The chosen values of K and R based on cross-validation.
`results.mean`	Mean squared error for all values of K and R considered.
`results.se`	Standard error of the sum of squared error for all values of K and R considered.

Author(s)

Kean Ming Tan and Daniela Witten

References

KM Tan and D Witten (2014) Sparse biclustering of transposable data. *Journal of Computational and Graphical Statistics* 23(4):985-1008.

D Witten, R Tibshirani, and T Hastie (2009) A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis, *Biostatistics* 10(3), 515–534.

See Also

`sparseBC` `sparseBC.BIC`

Examples

```
########### Create data matrix with K=2 R=4 row and column clusters
#k <- 2
#r <- 4
#n <- 200
#p <- 200
#  mus<-runif(k*r,-3,3)
#  mus<-matrix(c(mus),nrow=k,ncol=r,byrow=FALSE)
#  truthCs<-sample(1:k,n,rep=TRUE)
#  truthDs<-sample(1:r,p,rep=TRUE)
#  x<-matrix(rnorm(n*p,mean=0,sd=2),nrow=n,ncol=p)
#  for(i in 1:max(truthCs)){
#     for(j in 1:max(truthDs)){
#         x[truthCs==i, truthDs==j] <- x[truthCs==i, truthDs==j] + mus[i,j]
#    }
#  }
#  x<-x-mean(x)

# Example is commented out for short run-time
############ Perform sparseBC.choosekr to choose the number of row and column clusters
#sparseBC.choosekr(x,1:5,1:5,0,0.2)$estimated_kr
```

`summary.sparseBC` *Plot an object of class* `sparseBC` *or* `matrixBC`

Description

This function provides some information for an object `sparseBC` or `matrixBC`

Usage

```
## S3 method for class 'sparseBC'
summary(object, ...)
```

Arguments

`object`	an object of class `sparseBC` or `link{matrixBC}`
`...`	any other arguments passed to `print`.

Details

Some information for an object sparseBC. (1) Cluster labels for the rows/observations. (2) Cluster labels for the columns/features. (3) The estimated bicluster mean for each cluster.

Author(s)

Kean Ming Tan and Daniela Witten

References

#

See Also

`sparseBC` `matrixBC` `image.sparseBC` `sparseBC.BIC` `sparseBC.choosekr`

Examples

```
# See example in sparseBC
```

Index