R_CS_general_cov.R tries to fit penalized model based likelihood to the data assuming unconstrained cluster-specific covariance matrix for different clusters.

- The function is run using the command:

\[
\text{CS\_general\_cov}(Y,Y_{\text{tune}},n,n_{\text{tune}},k,k_0,\text{TRUE\_INDEX},\text{BIC},\text{num\_cluster},\lambda,\rho)
\]

where each argument means:

Therefore when knowing which variables are informative, the user should move those to the left most k0 columns of Y and Y_{tune}.

The function will return several R objects, which can be assigned to a variable. For example, with all default options, to save the results in the variable out, type the command:

\[
\text{out} = \text{CS\_general\_cov}(Y,Y_{\text{tune}},n,n_{\text{tune}},k,k_0,\text{TRUE\_INDEX},\text{BIC},\text{num\_cluster},\lambda,\rho,\text{MAX\_iter},\text{threshold},\text{MAX\_iter\_glasso},\text{threshold\_glasso})
\]

To see the results, use the “$” operator (VariableName$ObjectName). \text{CS\_general\_cov()} returns the following objects:
Y Training data of size $n \times k$, with each row represents one observation.

$Y_{\text{tune}}$ Tuning data of size $n_{\text{tune}} \times k$

n Number of observations for training data

$n_{\text{tune}}$ Number of observations for tuning data

k Dimension for both training and tuning data

$k_0$ First $k_0$ variables are informative; if this quantity is unknown, then $z_1$ and $z_2$ in the output should be ignored

$TRUE_{\text{INDEX}}$ The index showing the true group membership of each observation; if this quantity is unknown, then $RI$ and $aRI$ in the output should be ignored

BIC Boolean variable, if TRUE then select number of cluster, penalty parameters based on BIC, otherwise, use tuning data to select them. Default is FALSE

num_cluster A vector containing number of clusters to be considered. Default is from 1 to 10

lambda A vector containing penalty coefficients of the mean vector. Default is from 1 to 20, with step size 1

rou A vector containing penalty coefficients for off-diagonal elements in covariance matrix. Default is from 10 to 100, with step size 10

MAX_iter Maximum iteration allowed the EM algorithm. Default is 100

threshold Threshold for convergence. Default is 0.0001

MAX_iter_glasso Maximum iteration allowed in glasso. Default is 100

threshold_glasso Threshold for convergence in glasso. Default is 0.0001

Example:
In this example, we first generate a dataset consist of two clusters, with different general covariance structure and with different mean vector in the first 21 variables, while the remaining 279 variables are noninformative, i.e simulated from standard Normal distribution. The details of how to generate the data is shown below. After data generation, call the function and save the results:

```r
out = CS_general_cov(Y,Y_tune,n,n_tune,k,k0,TRUE_INDEX,BIC=F,
num_cluster=seq(1:5),lambda=seq(1,20,by=1),rou=seq(10,100,by=10),
MAX_iter=100,threshold=1e-4,MAX_iter_glasso=100,threshold_glasso=1e-4)
```
**num_cluster**  
optimal number of clusters

**lambda**  
optimal penalty parameter on the mean structure

**rou**  
optimal penalty parameter on off diagonal elements of covariance matrices

**group_member**  
posterior probability for each observation belonging to each cluster

**z_1**  
number of estimated non-informative variables among truly informative variables

**z_2**  
number of estimated non-informative variables among truly non-informative variables

**RI**  
rand index between the estimated group membership and TRUE_INDEX

**aRI**  
adjusted rand index between the estimated group membership and TRUE_INDEX

Then to see the output:

```r
> out$num_cluster
[1] 2
> out$lambda
[1] 8
> out$rou
[1] 30
> out$z_1
[1] 0
> out$z_2
[1] 262
> out$RI
[1] 1
> out$aRI
[1] 1
> out$group_member

[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[75] 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```
The entire example programs is as follows:

```r
code
source("R_CS_diag_cov.R")

# parameters
set.seed(2)
k0<-21  # number of attributes defining the clustering
k<- 300  # number of attributes total
n<- 100  # total number of objects
n0<-20  # number of objects in the small cluster
n_tune <-n*1  # total number of tuning objects
n0_tune <-n0*1  # number of tuning objects in the smaller cluster
nsim=1  # number of simulated data
u1=0.0  # the mean of the first cluster
rou_1=0.3  # AR(1) with rou=rou_1
rou_2=0.2  # AR(1) with rou=rou_2
u2=0.0  # the mean of one cluster other than the other mean cluster
du=2
du_tune=du

generate data
TRUE_INDEX <- c(rep(1,(n-n0)),rep(2,n0))
Y <- matrix(0,n,k)
Y_tune <- matrix(0,n,k)

cov_matrix_1 <- diag(1,k,k)
for(i in 1:k0){for(j in 1:k0){
cov_matrix_1[i,j] <- rou_1^abs(i-j) }}
```

4
cov_matrix_2 <- diag(1,k,k)
for(i in 1:k0){for(j in 1:k0){
cov_matrix_2[i,j] <- rou_2^abs(i-j) }
}

#
mu_1 <- c(rep(u1,k0),rep(u1,k-k0))
mu_2 <- c(rep(u1+du,k0),rep(u1,k-k0))
for(i in 1:(n-n0))
Y[i,] <- rmvnorm(1,mean=mu_1,sigma=cov_matrix_1)
for(i in (n-n0+1):n)
Y[i,] <- rmvnorm(1,mean=mu_2,sigma=cov_matrix_2)
#
mu_1 <- c(rep(u1,k0),rep(u1,k-k0))
mu_2 <- c(rep(u1+du_tune,k0),rep(u1,k-k0))
for(i in 1:(n-n0))
Y_tune[i,] <- rmvnorm(1,mean=mu_1,sigma=cov_matrix_1)
for(i in (n-n0+1):n)
Y_tune[i,] <- rmvnorm(1,mean=mu_2,sigma=cov_matrix_2)

########################################################################
out <- CS_general_cov(Y,Y_tune,n,n_tune,k,k0,TRUE_INDEX,BIC=F,
um_cluster=seq(1:5),lambda=seq(1,20,by=1),rou=seq(10,100,by=10),
MAX_iter=100,threshold=1e-4,MAX_iter_glasso=100,threshold_glasso=1e-4)
out$num_cluster
out$lambda
out$rou
out$z_1
out$z_2
out$RI
out$aRI
out$group_member