Package 'DNKF'

November 25, 2023

Title Differential Network Knockoff Filter

Version 2.0

Description Differential network ananlyse using Knockoff.

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Depends foreach

Imports clime,
DensParcorr,
doParallel,
glmnet,
glmnetUtils,
knockoff,
methods,
robustbase

R topics documented:

`DensPcorr` *Dens-based approach for precision matrix estimation.*

Description

Dens-based approach for precision matrix estimation.

Usage

```
DensPcorr(
  data,
  select = FALSE,
  dens.level = 0.5,
  plateau.thresh = 0.01,
  Parcorr.est = NULL,
  lambda = NULL
)
```

Arguments

data	Input data matrix of size n (observations) times p (variables).
select	Whether to conduct the Dens-based selection. If FALSE, output will only contain the estimated partial correlation list and precision matrix list corresponding to the default tuning parameter series ranging from 1e-8 to 0.6. If TRUE, the ouput will include the previous results and the selected partial correlation matrix and percision matrix corresponding to the specified density level. Default is FALSE.
dens.level	Specify the density level in Dens-based tuning parameter selection method (0<dens.level<1). This option is valid only when select=TRUE.
plateau.thresh	The criterion to select the plateau.
Parcorr.est	Previous output from DensPcorr function.
lambda	The tuning parameters for estimating the precision matrix ranging from 0 to 1.

Details

This function implements the statistical method proposed in Wang et al. (2016). See Rpackage "DensParcorr".

Value

An R list containing the following terms:

selected.precision
: Selected Precision matrix corresponding to dens.level.

selected.lambda
: Selected tuning parameter corresponding to dens.level.

DNKF.filter *Differential Network Knockoff Filter*

Description

Differential Network Knockoff Filter

Usage

```
DNKF.filter(
  data1,
  data2,
  Zv1 = NULL,
  Zv2 = NULL,
  screen_ratio = 0.5,
  B = 10,
  fdr = 0.1,
  offset = 0,
  cores
)
```

Arguments

data1	One group data. p-q-n1 array or A list of length n1, each of which is p-by-q spatial-temporal data matrix. n1 denotes the number of samples, p denotes the spatial dimension, q denotes the temporal dimension.
data2	Another group data. p-q-n2 array or A list of length n2, each of which is p-by-q spatial-temporal data matrix. n2 denotes the number of samples, p denotes the spatial dimension, q denotes the temporal dimension.
Zv1	A n1*v covariates matrix corresponding to data1.
Zv2	A n2*v covariates matrix corresponding to data2.
screen_ratio	Screen ratio. When screening, the number of screen is $ratio \cdot 0.5 \cdot p \cdot (p-1)$, (default: 0.5).
B	Times of generating knockoff variables (default: 10)
fdr	target false discovery rate (default: 0.1).
offset	either 0 or 1 (default: 0).
cores	Number of cores used in parallel (default: Number of all cores of the computer).

Value

Differential network matrix. The non-zero elements of the matrix represent the estimated difference of partial correlation coefficients

Examples

```
p <- 6; q <- 20; n <- 15
set.seed(2023)
Theta <- diag(p)
Theta[1,2] <- Theta[1,3] <- Theta[4,5] <- Theta[4,6] <- 1
Theta <- t(Theta)+Theta
diag(Theta) <- 1
omega1 <- Theta * sample(c(-1, 1), p * p, replace = TRUE) * runif(p * p, 0.4, 0.5)
omega1[lower.tri(omega1, diag = TRUE)] <- 0
omega1 <- as.matrix(omega1)
omega1 <- omega1 + t(omega1)
diag(omega1) <- abs(min(eigen(omega1)$values)) + 1
sigma1 <- cov2cor(solve(omega1))
omega1 <- solve(sigma1)
omega1[abs(omega1)<10^-4] <- 0
```

```
omega2 <- omega1
omega2[1:(p/2),1:(p/2)] <- -1*omega2[1:(p/2),1:(p/2)]
diag(omega2) <- diag(omega1)
sigma2 <- solve(omega2)
delta <- cov2cor(omega1)-cov2cor(omega2)
sigmaT1 <- 0.4^abs(outer(1:q,1:q,"-"))
sigmaT2 <- 0.5^abs(outer(1:q,1:q,"-"))

rmatnorm <- function(n,mean,sigmaS,sigmaT){
  nr <- nrow(sigmaS)
  nc <- ncol(sigmaT)
  R <- chol(sigmaT, pivot = TRUE)
  R <- R[, order(attr(R, "pivot"))]
  Q <- chol(sigmaS, pivot = TRUE)
  Q <- Q[, order(attr(Q, "pivot"))]
  mat = array(dim = c(nr, nc, n))
  for (i in 1:n) {
   mat[, , i] = mean + t(Q) %*% matrix(rnorm(nr * nc), nrow = nr) %*% R
  }
  mat
}

X1 <- rmatnorm(n,mean=matrix(0,p,q),sigmaS=sigma1,sigmaT=sigmaT1)
X2 <- rmatnorm(n,mean=matrix(0,p,q),sigmaS=sigma2,sigmaT=sigmaT2)

result <- DNKF.filter(data1 = X1, data2 = X2,  B = 4, cores = 2)
result
```

`stat.glmnet_coefdiff_bin` *Importance statistics based on a GLM with cross-validation*

Description

Importance statistics based on a GLM with cross-validation

Usage

```
stat.glmnet_coefdiff_bin(X, X_k, Zv, y, family = "binomial", ...)
```

Arguments

`X`	n-by-p matrix of original variables.
`X_k`	n-by-p matrix of knockoff variables.
`Zv`	a vector or matrix, containing the covariates.
`y`	vector of length n, containing the response variables.
`family`	response type.The default response family is 'binomial', for a logistic regression model.
`...`	Other arguments that can be passed to glmnet.

Value

A vector of statistics W of length p.

Index