Package: PCGII (via r-universe)

September 11, 2024

Type Package

Title Partial Correlation Graph with Information Incorporation

Version 1.1.2

Author Hao Wang [aut, cre], Yumou Qiu [aut], Peng Liu [aut]

Maintainer Hao Wang <haydo.wang@outlook.com>

Description Large-scale gene expression studies allow gene network construction to uncover associations among genes. This package is developed for estimating and testing partial correlation graphs with prior information incorporated.

License MIT + file LICENSE

URL https://haowang47.github.io/PCGII/

Depends R(>= 4.3.0)

Imports stats, corpcor (>= 1.6.10), glmnet, igraph (>= 1.5.0.1), Matrix, dplyr (>= 1.1.4)

Suggests mvtnorm, tidyverse, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Language en-US

Repository https://haowang47.r-universe.dev

RemoteUrl https://github.com/haowang47/pcgii

RemoteRef HEAD

RemoteSha d5212e388e5c3673f4087a722d5e93fd25539245

Contents

`clevel` *Get the estimated partial correlation graph without information incorporation*

Description

clevel() is the function to apply the method originally proposed in paper "Qiu, Y., & Zhou, X. H. (2020). Estimating c-level partial correlation graphs with application to brain imaging". It is used to get the estimated partial correlation graph without information incorporation. Remark: mathematical standardization will be automatically done within the function.

Usage

```
clevel(df, lambda)
```

Arguments

`df`	The main expression dataset, an n by p matrix, in which each row corresponds to a sample and each column represents expression/abundance of an omics feature.
`lambda`	The regularization parameter, used in the node-wise regression. If missing, default lambda will be used which is at the order of sqrt(2*log(p)/n).

Value

A list. The list contains estimated partial correlation matrix (Est), sparse partial correlation estimation matrix with threshold (EstThresh), estimated kappa (kappa), estimated test statistics matrix of partial correlations (tscore), sample size (n) and number of nodes (p).

Examples

```
library(PCGII)
library(corpcor)
library(glmnet)
library(igraph)
library(Matrix)
library(mvtnorm)
```

```
# Simulating data
set.seed(1234567)
n=50 # sample size
p=30 # number of nodes

Omega=make_random_precision_mat(eta=.01, p=p)

# population covariance matrix, which is used to generate data
Sigma=solve(Omega)
# simulate expression data
X = rmvnorm(n = n, sigma = Sigma)

lam=2*sqrt(log(p)/n) ## fixed lambda

CLEVEL_out=clevel(df=X, lambda = lam)
inference_out=inference(list=CLEVEL_out)
diag(inference_out)=0
net=graph_from_adjacency_matrix(inference_out, mode = "undirected")
   plot(net,
        vertex.size=4,
        vertex.label.dist=0.5,
        vertex.color="red",
        edge.arrow.size=0.5,
        layout=layout_in_circle(net))
```

`inference` *Conduct simultaneous inference of estimated partial correlations*

Description

Inference() is the function to conduct simultaneous inference of estimated partial correlations.

Usage

```
inference(list, alpha = 0.05)
```

Arguments

`list`	A list returned by either 'PCGII()' or 'clevel()'.
`alpha`	A pre-determined False Discovery Rate. Nominal FDR is set at 0.05 by default.

Value

An adjacency matrix of significant partial correlations.

Examples

```
library(igraph)
library(PCGII)
library(mvtnorm)
# Simulating data
set.seed(1234567)
n=50 # sample size
p=30 # number of nodes

Omega=make_random_precision_mat(eta=.01, p=p)

# population covariance matrix, which is used to generate data
Sigma=solve(Omega)
# simulate expression data
X = rmvnorm(n = n, sigma = Sigma)

lam=2*sqrt(log(p)/n) ## fixed lambda

# directed prior network
prior_set=as.data.frame(matrix(data=c(5,6, 28,24), nrow=2, ncol=2, byrow = TRUE))
colnames(prior_set)=c("row", "col")
prior_set=undirected_prior(prior_set)
PCGII_out=PCGII(df=X, prior=prior_set, lambda = lam)
inference_out=inference(list=PCGII_out)
diag(inference_out)=0
net=graph_from_adjacency_matrix(inference_out, mode = "undirected")
   plot(net, vertex.size=4,
        vertex.label.dist=0.5,
        vertex.color="red",
        edge.arrow.size=0.5,
        layout=layout_in_circle(net))
```

`makeBlockDiag` *Generate block-diagonal matrix of size p by p*

Description

A utility function generates block-diagonal matrix of size p by p with blocks B1, B2, ..., Bk. Each block matrix is of size blocksize by blocksize. The off-diagonal elements in block matrix are generated from uniform (min.beta, max.beta). The diagonal elements in block matrix are generated from uniform (1, 1.25).

Usage

```
makeBlockDiag(blocksize = 4, p = 20, min.beta = 0.3, max.beta = 0.9)
```

Arguments

blocksize	A positive integer, the dimension of the block matrix. Note, 'blocksize' has to be a factor of 'p'.
p	A positive integer, the size of the block-diagonal matrix.
min.beta	A positive number, lower limits of the uniform distribution.
max.beta	A positive number, upper limits of the uniform distribution.

Value

A block-diagonal matrix of size 'p' by 'p'.

Examples

```
mat = makeBlockDiag(blocksize=4, p=20)
```

---

make_random_precision_mat

Generate unstructured/random network skeleton and simulates corresponding precision matrix

---

Description

A utility function generates unstructured/random network skeleton and simulates corresponding precision matrix. The non-zero elements of the precision matrix are generated randomly from a uniform distribution with parameters (-upper, -lower) UNION (lower, upper).

Usage

```
make_random_precision_mat(
  eta = 0.01,
  p = 20,
  lower = 0.2,
  upper = 0.5,
  diag = 0.1
)
```

Arguments

eta	A number between 0 and 1, the probability for drawing an edge between two arbitrary vertices, i.e. the sparsity of the network.
p	A positive integer, the number of vertices.
lower	A positive number, lower limits of the uniform distribution.
upper	A positive number, upper limits of the uniform distribution.
diag	A small positive number to be added to diagonal elements, which guarantees the precision matrix is positive definite.

Value

A precision matrix of size p by p.

Examples

```
Omega = make_random_precision_mat(eta=.2, p=10)
```

`make_sf_precision_mat` *Generate scale-free network skeleton and simulates corresponding precision matrix*

Description

A utility function generates scale-free network skeleton and simulates corresponding precision matrix. The non-zero elements of the precision matrix are generated randomly from a uniform distribution with parameters (-upper, -lower) UNION (lower, upper).

Usage

```
make_sf_precision_mat(
  e = 1,
  power = 1,
  p = 20,
  lower = 0.2,
  upper = 0.5,
  diag = 0.1
)
```

Arguments

`e`	Numeric constant, the number of edges to add in each time step, see sample_pa().
`power`	Numeric constant, the power of the preferential attachment for scale-free network, the default is 1, , see sample_pa().
`p`	A positive integer, the number of vertices.
`lower`	A positive number, lower limits of the uniform distribution.
`upper`	A positive number, upper limits of the uniform distribution.
`diag`	A small positive number to be added to diagonal elements, which guarantees the precision matrix is positive definite.

Value

A precision matrix of size p by p.

Examples

```
Omega = make_sf_precision_mat(e=1, p=10)
```

PCGII *Get the estimated partial correlation graph with information incorporation*

Description

PCGII() is the function to apply the proposed method to get the estimated partial correlation graph with information incorporation. Remark: mathematical standardization will be automatically done within the function.

Usage

```
PCGII(df, prior, lambda)
```

Arguments

df	The main expression dataset, an n by p matrix, in which each row corresponds to a sample and each column represents expression/abundance of an omics feature.
prior	The prior set, a k by 2 dataframe, in which each row corresponds to a pair of nodes (any omics features) that are connected under prior belief. Note, prior input has to be dataframe.
lambda	The regularization parameter, used in the node-wise regression. If missing, default lambda will be used which is at the order of sqrt(2*log(p)/n).

Value

A list. The list contains estimated partial correlation matrix (Est), sparse partial correlation estimation matrix with threshold (EstThresh), estimated kappa (kappa), estimated test statistics matrix of partial correlations (tscore), sample size (n) and number of nodes (p).

Examples

```
library(PCGII)
library(corpcor)
library(glmnet)
library(igraph)
library(Matrix)
library(mvtnorm)
# Simulating data
set.seed(1234567)
n=50 # sample size
p=30 # number of nodes

Omega=make_random_precision_mat(eta=.01, p=p)

# population covariance matrix, which is used to generate data
Sigma=solve(Omega)
# simulate expression data
```

```
X = rmvnorm(n = n, sigma = Sigma)

lam=2*sqrt(log(p)/n) ## fixed lambda

# directed prior network
prior_set=as.data.frame(matrix(data=c(5,6, 28,24), nrow=2, ncol=2, byrow = TRUE))
colnames(prior_set)=c("row", "col")
prior_set=undirected_prior(prior_set)
PCGII_out=PCGII(df=X, prior=prior_set, lambda = lam)
inference_out=inference(list=PCGII_out)
diag(inference_out)=0
net=graph_from_adjacency_matrix(inference_out, mode = "undirected")
   plot(net, vertex.size=4,
        vertex.label.dist=0.5,
        vertex.color="red",
        edge.arrow.size=0.5,
        layout=layout_in_circle(net))
```

`sigs2mat` *Utility function for PCGII inference results*

Description

A utility function takes PCGII inference results as input and generates an adjacency matrix corresponding to the significant partial correlations

Usage

```
sigs2mat(sigs, P)
```

Arguments

`sigs`	A dataframe of locations (row, col) of selected edges.
`P`	A number, the number of nodes in the network.

Value

A matrix of size P*(P-1)/2, with 0, 1.

Examples

```
edges=cbind.data.frame(row=c(1,2,3,1,6,2,1,6,1,4),
                       col=c(2,1,1,3,2,6,6,1,4,1)) # five edges
sigs2mat(sigs = edges, P = 6)
```

undirected_prior *Pre-process the input prior set to ensure the input prior set corresponds to an undirected prior network*

Description

An utility function to pre-process the input prior set. This function will ensure the input prior set corresponds to an undirected prior network. If the prior network is believed to be directed, no pre-processing of the prior set is needed. Remark: this function is not necessary. Prior set should be considered carefully before running the network analysis. If the prior network connections are believed to be undirected while the prior set only includes one way connections for simplicity, this function will duplicate the connections and swap the direction automatically.

Usage

```
undirected_prior(prior)
```

Arguments

prior	A k by 2 data.frame of prior set, in which each row corresponds to a pair of nodes (any omics features) that are connected under prior belief

Value

A 2-column data.frame of pre-processed prior set, in which the connection between any pair of nodes is undirected.

Examples

```
prior=as.data.frame(matrix(c(1,2,1,5,1,10), ncol=2, byrow=TRUE))
## a prior set of 3 connections (1-2, 1-3, 1-10)
colnames(prior)=c("row", "col")
undirected=undirected_prior(prior)
```

Index