Package ‘BANFF’

October 12, 2015

Type Package

Title Bayesian Network Feature Finder

Version 1.0

Description Provides a full package of posterior inference, model comparison, and graphical illustration of model fitting. A parallel computing algorithm for the Markov chain Monte Carlo (MCMC) based posterior inference and an Expectation-Maximization (EM) based algorithm for posterior approximation are developed, both of which greatly reduce the computational time for model inference.

Depends R (&gt;= 3.0.3), foreach, doParallel

License GPL-2

Imports DPpackage, igraph, mclust, pscl, tmvtnorm, network, coda, graphics, methods, stats, utils

Author Zhou Lan, Yize Zhao, Jian Kang, Tianwei Yu

Maintainer Zhou Lan &lt;zlan6@gatech.edu&gt;

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-12 17:03:09

R topics documented:

DPM.HODC ........................................ 2
EM.HODC ........................................... 4
Grid.Adjmatrix.Transfer ............................. 5
HyperPara.Select ................................. 6
LikelihoodHistory ................................ 7
Networks.Fast .................................... 8
Networks.STD ................................... 11
plot.Networks.Fast ................................ 13
plot.Networks.STD ................................ 14
Plot.Subnetwork ................................. 16
Subnetwork.Select ............................... 16
summary.Networks.Fast ........................... 17
Hierarchical Ordered Density Clustering (HODC) for Dirichlet Process Mixture Model Fitting

Description

This function implements the HODC algorithm for posterior density samples by a Dirichlet process mixture (DPM) of normals model which is implemented by function “DPdensity” in R package DPPackages.

Usage

```r
DPM.HODC(v, pvalue, 
DPM.mcmc=list(nburn=2000,nsave=1, 
nskip=0,ndisplay=10), 
DPM.prior=list(a0=2,b0=1,m2=rep(0,1),
s2=diag(100000,1),
psiinv2=solve(diag(0.5,1)),
nu1=4,nu2=4,tau1=1,tau2=100))
```

Arguments

- `pvalue` a vector of p-values obtained from large scale statistical hypothesis testing
- `v` the number of posterior sample saved
- `DPM.mcmc` a list giving the MCMC parameters for DPM fitting; see the argument `mcmc` of function `DPdensity()` in DPPackage for details; the default setting is `DPM.mcmc=list(nburn=2000,nsave=1,nskip=0,ndisplay=10)`
- `DPM.prior` a list giving the prior information; see the argument `prior` of function `DPdensity()` in DPPackage for details; the default setting is `prior2`

Details

This function calls `DPdensity` to estimate the marginal density of the testing statistics `r`, converted from p-values, using a mixture of normal densities without incorporating the network information. Furthermore, it implements the HODC algorithm to classify density components into two clusters. We refer to them as the unimportant cluster and the important cluster, where the important cluster has a larger mean than the unimportant cluster.
Value

- *mean* a list containing posterior samples for the mean of unimportant and important clusters
  - *mu0* a vector of length "v" containing posterior samples for the mean of the unimportant cluster
  - *mu1* a vector of length "v" containing posterior samples for the mean of the important cluster

- *variance* a list containing posterior samples of the variance of the unimportant and the important clusters
  - *var0* a vector of length "v" containing posterior samples for the variance of the unimportant cluster
  - *var1* a vector of length "v" containing posterior samples for the variance of the important cluster

- *probability* a list containing the probabilities of unimportant and important clusters
  - *pro0* a vector of length "v" containing posterior samples for the probability of the unimportant cluster
  - *pro1* a vector of length "v" containing posterior samples for the probability of the important cluster

- *classification* a binary (0/1) matrix of dimension "v" by length(pvalue) containing posterior samples for two cluster classification results

Author(s)

Zhou Lan, Jian Kang, Tianwei Yu and Yize Zhao

Department of Biostatistics and Bioinformatics, Emory University

References


Zhou Lan, Jian Kang, Tianwei Yu, Yize Zhao, BANFF: an R package for network identifications via Bayesian nonparametric mixture models, working paper.

Examples

```r
# random make the density
rstat=c(rnorm(50,mean=1),rnorm(50,mean=2),rnorm(100,mean=4),
       rnorm(100,mean=8))
# transformed into pvalue
pvalue=pnorm(-rstat)
DPMHODC=DPM.HODC(v=5,pvalue)
```
EM.HODC

Hierarchical Ordered Density Clustering (HODC) for Finite Mixture Model Fitting

Description

This function implements the HODC algorithm for density estimates by a finite mixture of normals model which is implemented by function \texttt{mclust} in R package \texttt{mclust}.

Usage

\texttt{EM.HODC(pvalue)}

Arguments

\texttt{pvalue} a vector of p-values obtained from large scale statistical hypothesis testing

Details

This function calls \texttt{mclust} to estimate the marginal density of the testing statistics \( r \), converted from p-values, using a mixture of normal densities without incorporating the network information. Furthermore, it implements the HODC algorithm to classify density components into two clusters. We refer to them as the unimportant cluster and the important cluster, where the important cluster has a larger mean than the unimportant cluster.

Value

a list of density clustering results by the HODC algorithm

mean mean estimates of unimportant and important clusters

variance variance estimates of unimportant and important clusters

pro probability estimates of unimportant and important clusters

classification classifications configurations

Author(s)

Zhou Lan, Jian Kang, Tianwei Yu and Yize Zhao

References


Zhou Lan, Jian Kang, Tianwei Yu, Yize Zhao, BANFF: an R package for network identifications via Bayesian nonparametric mixture models, working paper.
Grid.Adjmatrix.Transfer

Examples

```r
rstat=c(rnorm(50,mean=1),rnorm(50,mean=2),rnorm(100,mean=4),rnorm(100,mean=8))
pvalue=pfnorm(-rstat)
EMHODC=EM.HODC(pvalue)
```

Description

This function is a supplementary function for transferring grid coordinates into adjacency matrix. The criteria for defining connected and disconnected: for any two vectors, if euclidean distance is less than a certain value, we consider these two vectors as connected, otherwise, disconnected. The purpose of this function is to quickly transform image data, such as brain image data, into adjacency matrix.

Usage

```r
Grid.Adjmatrix.Transfer(grid, euclidean.dist=1)
```

Arguments

- `grid`: a x by y matrix or data.frame, representing y vectors with dimension of x row binded (`rbind`) together.
- `euclidean.dist`: a number representing the maximum euclidean distance to be considered as connected. The default setting is 1.

Value

A y by y binary (0/1) adjacency matrix, where y is the row of the argument `grid`.

Note

Although in real world, the grid coordinates are less than 3-dimension, however, this function is still applicable for vectors larger than 3

Examples

```r
# 3-D coordinates
x <- matrix(rnorm(3*10), nrow = 10)
# a 10x10 adjacency matrix returned for 'net'
# Distance larger than 1 is considered as connected
net<-Grid.Adjmatrix.Transfer(grid=x, euclidean.dist=1)
```
### Description

This function is aiming at selecting appropriate hyper parameters by Bayesian Model Averaging. This process of this function has already embedded in function `Networks.STD()` and `Networks.Fast()`. The purpose of this function is for having the selected hyper-parameters independently.

### Usage

```r
HyperPara.Select(net,pvalue,piall,rhoall,n=30)
```

### Arguments

- `net`: an "n" by "n" binary adjacent matrix (0/1) for the network configuration with `n=length(pvalue)`
- `pvalue`: a vector of p-values obtained from large scale statistical hypothesis testing
- `piall`: a vector of possible choices for "pi0" in an increasing order
- `rhoall`: a vector of possible choices of "rho0" and "rho1" in an increasing order
- `n`: a number of iterations you set for Bayesian Model Averaging. The default setting is 30, which is accord with the embedded inner process of in function `Networks.STD()` and `Networks.Fast()`. More iterations could be set if you have a desire to have more accurate parameters

### Value

- a list of selected parameter
  - `pi0` the selected pi0
  - `rho0` the selected rho0
  - `rho1` the selected rho1

### Author(s)

Zhou Lan, Jian Kang, Tianwei Yu and Yize Zhao
Department of Biostatistics and Bioinformatics, Emory University

### References

Zhou Lan, Jian Kang, Tianwei Yu, Yize Zhao, BANFF: an R package for network identifications via Bayesian nonparametric mixture models, working paper.
**Description**

This function is a supplementary function for calculating and plotting the history of log-Likelihood value for each iteration. This function is used for tracking the evolution of the MCMC chain obtained from functions Networks.STD() and Networks.Fast().

**Usage**

```r
LikelihoodHistory(Trace, pvalue, Status=FALSE, True.Node)
```

**Arguments**

- **Trace**
  - the trace (results of each iteration) generated by the functions Networks.STD() and Networks.Fast(). Please set as `Trace=total$trace` where `total` is the output of the two functions, Networks.STD() and Networks.Fast().

- **pvalue**
  - a vector of p-values obtained from large scale statistical hypothesis testing. It should be the same vector which you set for the argument `pvalue` in functions Networks.STD() and Networks.Fast().

- **Status**
  - a logical variable indicating whether you have the assumed true value. `TRUE` means you have the assumed true value for plotting and `FALSE`, otherwise. The default setting is `FALSE`.

- **True.Node**
  - a binary (0/1) vector providing the assumed information of each node. The length of vector should be equal to the length of the vector of `pvalue`.

**Value**

- a table indicating the history of the history of log-Likelihood value for each iteration.

**Note**

A plot describing the history of the history of log-Likelihood value for each iteration will be output. If you set `Status` and `True.Node`, a red line indicates the log-Likelihood value of the assumed status of each node.

**Author(s)**

Zhou Lan, Jian Kang, Tianwei Yu and Yize Zhao
Department of Biostatistics and Bioinformatics, Emory University
References


Zhou Lan, Jian Kang, Tianwei Yu, Yize Zhao, BANFF: an R package for network identifications via Bayesian nonparametric mixture models, working paper.

---

Networks.Fast

Bayesian Network Discovery using a Hybrid Fast Algorithm

Description

This function implements a hybrid fast algorithm to perform feature selection and sub-network discovery using a Bayesian nonparametric model based on Dirichlet process mixture models, finite mixture of normals model and the Ising model.

Usage

```r
Networks.Fast(pvalue, net, iter=5000, nburns=2000, algorithms=c("EM","DPM"), v=20, DPM.mcmc=list(nburn=2000, nsave=1, nskip=0, ndisplay=10), DPM.prior=list(a0=2, b0=1, m2=rep(0,1), s2=diag(100000,1), psiinv2=solve(diag(0.5,1)), nu1=4, nu2=4, tau1=1, tau2=100), DParallel=FALSE, n.cores=1, piall=c(0.8, 0.85, 0.9, 0.95), rhoall=c(1, 2, 5, 10, 15), show.steps=10, showlikelihood=FALSE, likelihood.frequency=100)
```

Arguments

- **pvalue**: a vector of p-values obtained from large scale statistical hypothesis testing
- **net**: an "n" by "n" binary (0/1) adjacent matrix of network configurations, where n=length(pvalue)
- **iter**: number of iterations; the default is 5000
- **nburns**: number of burn-in; the default is 2000
- **algorithms**: character taking value "EM" or "DPM" indicating the function to be used to obtain Finite Gaussian Mixture (FGM) estimates. It is recommended to choose "DPM" when the dimension of data is large, and to choose "EM" when the dimension is small.
- **v**: number of iterations set for DPM fitting. v is only valid when you choose algorithms as "DPdensity"
- **DPM.mcmc**: a list giving the MCMC a list giving the MCMC parameters for DPM fitting; see the argument mcmc of function DPdensity() in Dpackage for details; the default setting is DPM.mcmc=list(nburn=2000, nsave=1, nskip=0, ndisplay=10)
Details

This function implements a Bayesian nonparametric mixture model for feature selection incorporating network information (Zhao et al., 2014):

- \( r_i | g_i, \theta \sim N(\mu_{g_i}, \sigma_{g_i}) \),
- \( g_i | z_i=k, q_k \sim \text{Discrete}(a_k, q_k) \),
- \( \theta \sim G_0 \) for \( g \) in \( a_0 \),
- \( q_k \sim \text{Dirichlet}(\tau_k 1_{[L_k]/L_k}) \),
- \( \theta = \{ \theta_{g} \}_{g \in a_0 \text{ and } a_1} \)
- \( \theta_g = (\mu_g, \sigma_g) \)

where we define

Index: \( \mathbf{a}_0=(-L_0+1,-L_0+2,...,0) \), \( \mathbf{a}_1=(1,2,...,L_1) \) and the correspondent probability \( q_0=0, q_1=1_{[L_1]}, q_{L_0+2}, ..., q_0 \), \( q_1=0, q_2, ..., q_{L_1} \), according to the definition of \( \text{Discrete}(a_k, b_k) \), for example, \( \Pr(g_i=L_0+2)=q_{L_0+2} \).

Assumption In this algorithm, we assume that "important" features should have larger statistics comparing to "unimportant" ones without the loss of generality. In this regard, we set the restriction \( \mu_{g} < \mu_{g+1} \) for \( g=-L_0+1, -L_0+2,...,L_1 \).

This function implements the NET-DPM-3 Zhao et al.(2014). Please refer to the Appendix B.3 for more details.
Value

An object of class "Networks.Fast" containing a list of values, such as, the posterior samples of "z_i" and summary statistics.

trace an length(pvalue) by (iter-nburns) matrix

convergence MCMC Heidelberger and Welch convergence diagnostic

graph An igraph graph object of full network

statistics a list of summary statistics characterizing the posterior distribution of "z_i"

mean posterior mean for each feature

median posterior median for each feature

var posterior variance for each feature

quantile posterior quantiles for each feature

Author(s)

Zhou Lan, Jian Kang, Tianwei Yu and Yize Zhao

Department of Biostatistics and Bioinformatics, Emory University

References


Zhou Lan, Jian Kang, Tianwei Yu, Yize Zhao, BANFF: an R package for network identifications via Bayesian nonparametric mixture models, working paper.

Examples

###Gene Network discovery

###Generating Scale free Gene Network

library(igraph)

g <- barabasi.game(50, power=1, zero.appeal=1.5, directed = FALSE)

net <- as(get.adjacency(g, attr=NULL), "matrix")

###Random assign selected genes and make the signal intension as gaussian mixture

newz <- rep(c(1, 0, 0, 1, 0), 10)

Simnorm <- function(n){
  weight = c(0.4, 0.6)
  mu = c(0, 6)
  sigma = c(1, 0.5)
  z = sample(c(1, 2), size=n, prob=weight, replace=TRUE)
  r = rnorm(n, mean=mu[z], sd=sigma[z])
  return(r)
}

testcov <- 0

for(i in 1:50){
  if(newz[i] == 0){
    testcov[i] <- rnorm(1, mean=0, sd=1)
  } else {
    testcov[i] <- Simnorm(1)
  }
}
Bayesian Network Discovery using a Standard MCMC Algorithm

Description

This function implements the standard Markov chain Monte Carlo (MCMC) algorithm to perform feature selection and sub-network discovery using a Bayesian nonparametric model based on the Dirichlet process mixture (DPM) model and the Ising model.

Usage

Networks.STD(pvalue, net, iter=5000, nburns=2000,
               piall=c(0.75, 0.8, 0.85, 0.9), rhoall=c(0.5, 1, 5, 10, 15),
               status=FALSE, fit, show.steps=1, showlikelihood=FALSE, likelihood.frequency=100)

Arguments

- **pvalue**: a vector of p-values obtained from large scale statistical hypothesis testing
- **net**: an "n" by "n" binary adjacent matrix (0/1) for the network configuration with n=length(pvalue)
- **iter**: the number of iterations; the default is 5000
- **nburns**: the number of burn-in; the default is 2000
- **piall**: a vector of possible choices for "pi0" in an increasing order; the default value is c(0.75, 0.8, 0.85, 0.9)
- **rhoall**: a vector of possible choices of "rho0" and "rho1" in an increasing order; the default value is c(0.5, 1, 5, 10, 15)
- **status**: a logical variable indicating whether the you are initial an MCMC chain or resume a chain
- **fit**: a list provide the current values of parameters which are used for resume the chain
- **show.steps**: integer representing the frequency of the results of iterations presented, the default value is 1. The setting is invalid when trace=FALSE. The setting would not affect the data saved, only for printing
- **showlikelihood**: a logical variable indicating whether to show the log-likelihood value simultaneously. Set TRUE if show the log-likelihood value simultaneously, FALSE, otherwise. FALSE is the default setting
likelihood.frequency

A number representing the frequency showing the log-likelihood value simultaneously. For example, setting likelihood.frequency=100 means showing the log-likelihood value every 100 iterations. The default setting is 100 and it is recommended that do not set a small frequency because it slow down the MCMC chain updating.

Details

This function implements a Bayesian nonparametric mixture model for feature selection incorporating network information (Zhao et al., 2014):

- \( r_i | g_i, \theta \sim N(\mu_{g_i}, \sigma_{g_i}) \),
- \( g_i | z_i=k, q_k \sim \text{Discrete}(a_k, q_k) \),
- \( \theta \sim G_0k \) for \( g \) in \( a_k \),
- \( q_k \sim \text{Dirichlet}(\tau_{L_0}/L_k) \),
- \( \text{theta}=\{\theta_g\}_{g \in a_0 \text{ and } a_1} \)
- \( \theta_g=(\mu_g, \sigma_g) \)

Where we define

**Index** \( a_0=(-L_0+1,-L_0+2,...,0) \) , \( a_1=(1,2,...,L_1) \) and the correspondent probability \( q_0=(q_{-L_0+1}, q_{-L_0+2}, ..., q_0) \), \( q_1=(q_1, q_2, ..., q_{L_1}) \), according to the definition of \( \text{Discrete}(a_k, b_k) \), for example, \( \text{Pr}(g_i=L_0+2)=q_{-L_0+2} \).

**Assumption** In this algorithm, we assume that "important" features should have larger statistics comparing to "unimportant" ones without the loss of generality. In this regard, we set the restriction \( \mu_g<\mu_{g+1} \) for \( g=-L_0+1, -L_0+2,...,L_1 \).

This function implements the NET-DPM-1 Zhao et al.(2014). Please refer to the Appendix B.1 for more details.

Value

An object of class "Networks.STD" containing Bayesian information of the MCMC chain

- **trace** a matrix of dimension (iter-nburns) by length(pvalue) containing posterior samples of classification label "g_i"
- **convergence** MCMC Heidelberger and Welch convergence diagnostic
- **graph** An igraph graph object of full network
- **model** a list containing parameters of the MCMC chain for resuming the chain

Author(s)

Zhou Lan, Jian Kang, Tianwei Yu and Yize Zhao

Department of Biostatistics and Bioinformatics, Emory University
References
Zhou Lan, Jian Kang, Tianwei Yu, Yize Zhao, BANFF: an R package for network identifications via Bayesian nonparametric mixture models, working paper.

Examples

```r
library(igraph)
###Creating the network of 10X10 image
g <- graph.lattice(length=10,dim=2)
net=as(get.adjacency(g,attr=NULL),"matrix")##this is the input of argument 'net'
##Assign the signal elements with signal intention
##as normal distribution N(1,0.2). While noise is set as N(0,0.2)
newz=rep(0,100)
for (i in 1:7)
{
  newz[(i*10+3):(i*10+7)]=1
}

testcov<-0
for(i in 1:100){
  if(newz[i]==0){
    testcov[i]<-rnorm(1,mean=0, sd=0.2)
  }else{
    testcov[i]<-rnorm(1,mean=1, sd=0.2)
  }
}
##The profile of the image
image(matrix(testcov,10,10),col=gray(seq(0,1,length=255)))
##Transform the signals into pvalue form and begin identification
pvalue=pnorm(-testcov)
total=Networks.STD(pvalue,net,iter=3,nburns=1,
piall=c(0.8, 0.85, 0.9, 0.95),rhoall=c(0.5,1,5,10,15))
```

plot.Networks.Fast  

### Plotting Bayesian Network Discovery using a Hybrid Fast Algorithm

Description

plot method for class "Networks.Fast"

Usage

```r
## S3 method for class 'Networks.Fast'
plot(x, ...)
```
Arguments

- **x**: An object of class "Networks.Fast" resulting of a call to `Networks.Fast`.
  - Other arguments

See Also

- `Networks.Fast`.

Examples

```r
#library(igraph)
###Gene Network discovery
###Generating Scale free Gene Network
library(igraph)
g <- barabasi.game(50, power=1, zero.appeal=1.5,directed = FALSE)
net=as(get.adjacency(g,attr=“NULL”),”matrix”)
###Random assign selected genes and make the signal intension as gaussian mixture
newz=rep(c(1,0,0,1,0,10)
Simnorm=function(n){
weight = c(0.4, 0.6)
mu = c(8,6)
sigma = c(1,0.5)
z = sample(c(1,2),size=n, prob=weight,replace=TRUE)
r = rnorm(n,mean=mu[z],sd=sigma[z])
return(r)
}
testcov<-0
for(i in 1:50){
if(newz[i]==0){
  testcov[i]<-rnorm(1,mean=0,sd=1)
}else{
  testcov[i]<-Simnorm(1)
}
}
pvalue=pnorm(-testcov)
total1=Networks.Fast(pvalue,net,iter=5,nburns=2,
v=20,algorithms=“DPM”,DPparallel=FALSE,
piall=c(0.8, 0.85, 0.9, 0.95),rhoall=c(1, 2, 5, 10, 15))
}
```

```r
###Plotting the object of class "Networks.Fast"
plot(total1)
```
Usage

```r
## S3 method for class 'Networks.STD'
plot(x, ...)
```

Arguments

- `x` An object of class "Networks.STD" resulting of a call to `Networks.STD`.
- `...` other arguments

See Also

- `Networks.STD`

Examples

```r
library(igraph)

###Creating the network of 10X10 image

g <- graph.lattice(length=10,dim=2)
net<-(get.adjacency(g,attr=NULL),"matrix")
##this is the input of argument 'net'
##Assign the signal elements with signal intention
##as normal distribution N(1,0.2). While noise is set as N(0,0.2)
newz<-rep(0,100)
for (i in 3:7)
{
  newz[(i*10+3):(i*10+7)]=1
}
testcov<-0
for(i in 1:100){
  if(newz[i]==0){
    testcov[i]<-rnorm(1,mean=0, sd=0.2)
  }else{
    testcov[i]<-rnorm(1,mean=1, sd=0.2)
  }
}

###The profile of the image

image(matrix(testcov,10,10),col=gray(seq(0,1,length=255)))

###Transform the signals into pvalue form and begin identification

pvalue<-pnorm(-testcov)
total<-(Networks.STD(pvalue,net,iter=3, nburns=1,
piall=c(0.8, 0.85, 0.9, 0.95), rhoall=c(0.5,1,5,10,15))

###plot the object of class "Networks.STD"

plot(total)
```
Description

This is a supportive function for plotting the network feature according to the outputs of `network.NSTD` and `network.NFast`. If the mean of the marginal distribution of a certain node is larger than 0.5, we consider this node as selected.

Usage

`Plot.Subnetwork(net, trace)`

Arguments

- `net` an "n" by "n" binary (0/1) adjacent matrix of network configurations, where n=length(pvalue); same as the argument net in function `network.NSTD()` and `network.NFast()`
- `trace` the trace (results of each iteration) generated by the functions `networks.NSTD()` and `networks.NFast()`. Please set as `trace=total$trace` where total is the output of the two functions, `networks.NSTD()` and `networks.NFast()`

Value

A plot of the network feature selected

Examples

```
library(igraph)
g=barabasi.game(10, power=0.5)
net=as(get.adjacency(g),"matrix")
trace=matrix(c(rep(1,5),rep(0,5)),nrow=1)
Plot.Subnetwork(net,trace)
```

Description

This function is to Summarize the information of the sub networks selected by `network.NFast()` and `network.NSTD()`. If the marginal distribution of a certain node is larger than 0.5, we consider this node as selected. The aim of this function is to customize the process of summarizing.
Usage

Subnetwork.Select(net, trace, node.based=NULL, infinite=TRUE, steps=5)

Arguments

- `net`: an "n" by "n" binary (0/1) adjacent matrix of network configurations, where n=length(pvalue); same as the argument `net` in function `Network.STD` and `Network.Fast`
- `trace`: the trace (results of each iteration) generated by the functions `Networks.STD()` and `Networks.Fast()`. Please set as `trace=total$trace` where `total` is the output of the two functions, `Networks.STD()` and `Networks.Fast()`; you can also provide a binary 1 by "n" matrix representing the "important" (1) and "unimportant" (0) nodes
- `node.based`: a vector representing the orders of the nodes the sub network centered
- `infinite`: a logical variable indicating whether the expanding of sub network in a "infinite" pattern or not; `infinite=TRUE` if searching the "important" sub network based on one node without the limitation of maximum steps; `infinite=FALSE` if searching the sub network under a regulated way
- `steps`: a number representing the maximum steps for expanding the sub network

Value

A list of values containing the information of the sub networks selected by `Network.Fast()` and `Network.STD()`

- `eids`: a vector of order representing the selected nodes
- `adj`: a "m" by "m" binary adjacent matrix of the sub network configurations where n=length(eids)

Description

Summary method for class "Networks.Fast"

Usage

```r
## S3 method for class 'Networks.Fast'
summary(object, ...)
```

Arguments

- `...`: other arguments
See Also

Networks.Fast.

Examples

library(igraph)
###Gene Network discovery
#Generating Scale free Gene Network
library(igraph)
g <- barabasi.game(50, power=1, zero.appeal=1.5,directed = FALSE)
net=as(get.adjacency(g,attr=NULL),"matrix")
##Random assign selected genes and make the signal intension as gaussian mixture
newz=rep(c(1,0,0,1,0),10)
Simnorm=function(n){
  weight = c(0.4, 0.6)
  mu = c(8,6)
  sigma = c(1,0.5)
  z = sample(c(1,2),size=n, prob=weight,replace=TRUE)
  r = rnorm(n,mean=mu[z],sd=sigma[z])
  return(r)
}
testcov<-0
for(i in 1:50){
  if(newz[i]==0){
    testcov[i]<-rnorm(1,mean=0,sd=1)
  }else{
    testcov[i]<-Simnorm(1)
  }
}
pvalue=pnorm(-testcov)
total1=Networks.Fast(pvalue,net,iter=5,nburns=2,
v=20,algorithms="DPM",DPparallel=FALSE,
piall=c(0.8, 0.85, 0.9, 0.95),rhoall=c(1, 2, 5, 10, 15))

summary(total1)

summary.Networks.STD  Summary method for class "Networks.STD"

Usage

## S3 method for class 'Networks.STD'
summary(object, ...)
Arguments

- object: An object of class "Networks.STD" resulting of a call to Networks.STD.
- ... other arguments

See Also

Networks.STD.

Examples

library(igraph)
###Creating the network of 10X10 image
g <- graph.lattice(length=10,dim=2)
net=as(get.adjacency(g,attr=NULL),"matrix")##this is the input of argument 'net'
##Assign the signal elements with signal intention
##as normal distribution N(1,0.2). While noise is set as N(0,0.2)
newz=rep(0,100)
for (i in 3:7)
{  
  newz[(i*10+3):(i*10+7)]=1
}
testcov<0
for(i in 1:100){
  if(newz[i]==0){
    testcov[i]<-rnorm(1,mean=0,sd=0.2)
  }
}
}
##The profile of the image
image(matrix(testcov,10,10),col=gray(seq(0,1,length=255)))
##Transform the signals into pvalue form and begin identification
pvalue=pnorm(-testcov)
total=Networks.STD(pvalue,net,iter=3,nburns=1,
piall=c(0.8, 0.85, 0.9, 0.95),rhoall=c(0.5,1,5,10,15))

#####summary the object of class "Networks.STD"
summary(total)

SummaryAccuracy

Summary of the Accuracy of Node and Network selection

Description

This function is to summarize the accuracy of node and Network selection achieved by the functions Networks.STD and Networks.Fast. This function is designed to validate whether the settings of arguments are appropriate for one simulation study and provide insights of setting arguments for real data.
Summary Accuracy

Usage

```r
SummaryAccuracy(Trace, No.Sets,
Type.Accuracy=c("Node","Sub-network"), True.Node,
TruePositive.Net, FalsePositive.Net,
Type.Net.Accuracy=c("Marginal","Sample"), Tolerance)
```

Arguments

- **Trace**
  - the trace (results of each iteration) generated by the functions `Networks.STD()` and `Networks.Fast()`. For `Networks.STD()` and `Networks.Fast()`, set `Trace=total$trace` where `total` is the output of the two functions, `Networks.STD()` and `Networks.Fast()`. Use `rbind()` bind the inputs if you have multiple sets of results.

- **No.Sets**
  - number of sets of results you put in the argument Trace

- **Type.Accuracy**
  - character taking value "Node" or "Sub-network". "Node" represents checking the accuracy of each node; "Sub-network" represents checking the accuracy of network of interest.

- **True.Node**
  - a binary(0/1) vector providing the information of each node. The length of vector should be equal to the length of the vector of `pvalue`.

- **TruePositive.Net**
  - a vector representing the network of interest consisted of assumed positive nodes. For example, if you assume No.6, No.8 and No.9 nodes consist the network of interest, set `TruePositive.Net=c(6, 8, 9)`.

- **FalsePositive.Net**
  - a vector representing the assumed negative nodes which would lead false positive networking (larger network) if such nodes are assumed positive in the results. Such nodes should be connected to the network of interest and assumed negative. For example, if you assume No.3 and No.11 nodes would lead false positive networking (larger network), set `TruePositive.Net=c(3, 11)`.

- **Type.Net.Accuracy**
  - character taking value "Marginal" or "Sample". "Marginal" represents using marginal distribution of each data-set for checking accuracy, for example, if the probability (mean) of one node of a data-set is larger than 0.5, we consider it positive, otherwise, negative. Setting "Sample" represents checking accuracy through each iteration. For `Type.Accuracy="Node"`, only when `Type.Net.Accuracy="Sample"`, is valid.

- **Tolerance**
  - percentage of the Tolerance rate of accuracy. This argument is only valid when set `Type.Accuracy="Networking"`. Setting `Tolerance=0.9` means we consider selecting a correct networking selection if more than 0.9 positive nodes of networking or `TruePositive.Net` are considered as positive AND less than 0.1 negative nodes of networking or `FalsePositive.Net` are considered as positive.

Value

- a list of values containing the information of accuracy of nodes and network selection

  - **TPR.average** The Average of True Positive Rate
Summary Accuracy

**FPR.average** The Average of True False Rate

**FDR.average** The Average of False Positive Rate

**TPR** True Positive Rate of each node

**FPR** False Positive Rate of each node

**Author(s)**

Zhou Lan, Jian Kang, Tianwei Yu and Yize Zhao

Department of Biostatistics and Bioinformatics, Emory University

**References**


Zhou Lan, Jian Kang, Tianwei Yu, Yize Zhao, BANFF: an R package for network identifications via Bayesian nonparametric mixture models, working paper.

**Examples**

```r
###Gene Network discovery
#Generating Scale free Gene Network
library(igraph)
library(BANFF)
g <- barabasi.game(50, power=1, zero.appeal=1.5,directed = FALSE)
net<-as(get.adjacency(g,attr=NULL),"matrix")
#Random assign selected genes and make the signal intension as gaussian mixture
newz<-rep(c(1,0,0,1,0),10)
Simnorm=function(n){
    weight = c(0.4, 0.6)
    mu = c(8,6)
    sigma = c(1,0.5)
    z = sample(c(1,2),size=n, prob=weight,replace=TRUE)
    r = rnorm(n,mean=mu[z],sd=sigma[z])
    return(r)
}
testcov<-0
for(i in 1:50){
    if(newz[i]==0){
        testcov[i]<-rnorm(1,mean=0,sd=1)
    }else{
        testcov[i]<-Simnorm(1)
    }
}
pvalue=pnorm(-testcov)
total1=Networks.Fast(pvalue,net,iter=5,nburns=2,
v=20,algorithms="DPM",
piall=c(0.8, 0.85, 0.9, 0.95),rhoall=c(1, 2, 5, 10, 15))

###checking accuracy

###checking accuracy of nodes
SummaryAccuracy(Trace=total1$trace,NoSets=1,Type.Accuracy="Node",
True.Node=newz,Type.Net.Accuracy="Sample")
```
Index

*Topic \textasciitilde kwd1
  summary.Networks.Fast, 17
  summary.Networks.STD, 18

*Topic \textasciitilde kwd2
  summary.Networks.Fast, 17
  summary.Networks.STD, 18

DPM.HODC, 2
EM.HODC, 4
Grid.Adjmatrix.Transfer, 5
HyperPara.Select, 6
LikelihoodHistory, 7
Networks.Fast, 8, 14, 17, 18
Networks.STD, 11, 15, 19

plot.Networks.Fast, 13
plot.Networks.STD, 14
Plot.Subnetwork, 16

Subnetwork.Select, 16
summary.Networks.Fast, 17
summary.Networks.STD, 18
SummaryAccuracy, 19