Package ‘BANFF’

March 11, 2017

Type Package
Title Bayesian Network Feature Finder
Version 2.0
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 (>= 3.0.3)
License GPL-2
Imports DPpackage, igraph, mclust, GGally, truncnorm, graphics, stats, flexmix, grDevices, foreach, doParallel, coda, pscl, methods,
Author Zhuxuan Jin, Zhou Lan, Yize Zhao, Jian Kang, Tianwei Yu
Maintainer Tianwei Yu <tianwei.yu@emory.edu>
NeedsCompilation no
Repository CRAN
Date/Publication 2017-03-11 10:28:57
RoxygenNote 6.0.1

R topics documented:

BANFF2 ............................................................... 2
class.label .......................................................... 5
DPM.HODC ......................................................... 5
EM.HODC ........................................................... 6
miscloc .............................................................. 7
net ................................................................. 8
Networks.Fast ..................................................... 8
Networks.STD ..................................................... 11
SimulatedDataGenerator ......................................... 13
SummaryClassLabel ............................................... 15
test.stat .......................................................... 16
Bayesian nonparametric feature selection over large-scale networks with missing values

**Description**
Main function. Two steps: Given density specification, update selection indicator $z$ by Swendsen-Wang; Given selection indicator $z$, update density specification by DPM fitting.

**Usage**
```
BANFF2(net,test.stat,pvalue.stat=FALSE,candidate.z.set=c(-1,0,1),
seed.main=1024,na.action=c("NN","Bayes","na.remove"),niter.densupd=5,niter=10,
paras=list(tau=c(2,10,2),alpha=NULL,gamma=NULL,xi=NULL,beta=rep(10,3),
rho=c(1.003,0.479,0.988,0.000),pvec=c(0.15,0.7,0.15),densAcc=0.001,
null.quantile=c(0.25,0.75),null.method="bigGaussianModeSplit",
transitionMatrix.Z.11=0.6,miss.stat=2,min.node=5),
para.DPM=NULL,para.HODC=NULL,para.DMH=NULL)
```

**Arguments**
- **net**
  The adjacent matrix with 0/1 indicating "connected" or "not directly connected
- **test.stat**
  The observed test statistics. Missing values are represented as NAs. If they are pvalues, then the pvalue.stat should be T;
- **pvalue.stat**
  Logical. Whether test.stat is generated as pvalues or not. Default F.
- **candidate.z.set**
  Default is of three regulation type. Default=c(-1,0,1), 1=down-regulated, 2=not differentially expressed, 3=up-regulated.
- **seed.main**
  Set seed before iteration for generating reproducible results. Default=1024.
- **na.action**
  The method used to impute missing values. Can be "NN", "Bayes", or "na.remove".
- **niter.densupd**
  The total number of iterations for updating density. Default=5
- **niter**
  The total number of iterations for study. Default=10.
- **paras**
  A list contains hyper-parameters and other parameters used for preparations.
  - **niter.densupd**
    The iteration is from 1 to the maximum steps when we update density specification by DPM. Default=20.
  - **tau**
    A three-element vector. Default=c(2,10,2);
  - **alpha**
    A three-element vector. Default=NULL.
  - **gamma**
    A three-element vector. Default=NULL.
  - **xi**
    A three-element vector. Default=NULL.
  - **beta**
    A three-element vector. Default=rep(10,3).
  - **rho**
    A four-element vector. Default=c(1.003,0.479,0.988,0.000), indicating local smoothness for Potts prior. Note: the default value is calculated based on data(net) structure by DMH.
• pivec A three-element vector. Default=c(0.15,0.7,0.15). Contains prior knowledge globally about selection indicator z.
• densityAcc A number, need to specify precision for K-L integration when to use the numerical approximation. Default=0.001.
• null.quantile A two element vector representing lower quantile and upper quantile for calculating prior null density if not given by biologists. Default=c(0.25, 0.75).
• null.method A char. The method we used to estimate null density: "biGaussian"– EM algorithm for mixtures of two univariate normals; "biGaussianWith0Cutoff"– assume all negative test statistics forms one normal and all positive test statistics forms the other one normal. And proportion parameter is proportional to a number of observations each class; "biGaussianMean0"– null is formed by two half normals. "biGaussianModeSplit"– split data from median value, then flip each part to the other side to estimate normal distribution.
• transitionMatrix.Z.11 [1,1] element in transition matrix for z. Default=0.6.
• miss.stat impute NAs in test.test when apply Double Metropolis-Hastings sampler (DMH) to find hyperparameters: rho & pi.
• min.node The minimum number of nodes in each group.

para.DPM
A list object contains, if NULL, default value is used:
• niter default=10
• nsample default=10
• KLrange default=c(-6,6), usually we consider wider range than c(floor(min(test.stat,na.rm=TRUE)),ceiling(max(test.stat,na.rm=TRUE)))
• KLprecision default=0.001
• KLNullmethod default="biGaussianMean0"
• mcmc a list, default=list(nburn=10000,nsave=100,nskip=0,ndisplay=10000)
• prior a list, default=list(alpha=3,m1=rep(0,1),psiinv1=diag(0.5,1),nu1=4,tau1=1,tau2=100)

para.HODC
A list object contains, if NULL, default value is used:
• nsample default=10
• KLrange default=c(-6,6), usually we consider wider range than c(floor(min(test.stat,na.rm=TRUE)),ceiling(max(test.stat,na.rm=TRUE)))
• KLprecision default=0.001
• KLNullmethod default="biGaussianMean0",
• mcmc a list, default=list(nburn=1000,nsave=100,nskip=0,ndisplay=1000)
• prior a list, default is a list object where each of the element specify the prior used when fitting each density for class labels z. For each of the class, default parameters are the same, a list contains: alpha=3,m2=rep(0,1),s2=diag(100000,1),psiinv2=diag(temp.sdlist[1],1),nu1=4,nu2=4,tau1=1,tau2=100

para.DMH
If rho & pivec is not given, DMH is used for pre-calculating rho & pivec. Default is a list object contains:
• niter default=1000
• pistat default=c(0.25,0.5,0.25)
• pisd default=rep(0.03,3)
• rhostat default=c(1,0.5,1,0)
• rhosd default=rep(0.03,4)
Details

The fully Bayesian updating algorithm is executed as below:

- Input data r and graph \( G = \langle V, E \rangle \)
- Update \( z \mid \theta \) via Swendsen-Wang
- Update \( \theta \mid z \) via DPM Fitting

Value

A list:
- \texttt{initialValue} initial parameter list
- \texttt{zTrack} trace for \( z \)
- \texttt{FinalValue} final parameter list
- \texttt{iters} total iterations
- \texttt{rmisTrack} (if NAs in \texttt{test.statistics}) trace for \texttt{test.statistics} imputation. (only for those with NAs)

Examples

```r
# Not run:
# The simulation settings based on real gene network (takes time)
data("net")
data("test.stat")
res=BANFF2(net,test.stat,niter=300,na.action="NN")
res=BANFF2(net,pnorm(test.stat),pvalue.stat=TRUE,candidate.z.set=c(0,1),na.action="NN",niter=300,
paras=list(tau=c(2,10),alpha=NULL,gamma=NULL,xi=NULL,beta=rep(10,2),rho=c(1,0.5,0),
pivvec=c(0.2,0.8),densAcc=0.001,null.quantile=c(0.25,1),
nul.method="biGaussianModeSplit",transitionMatrix.Z11=0.6,miss.stat=2,min.node=5))

# A toy example
simdata=SimulatedDataGenerator(nnode=100,missing=TRUE,missrate=0.1,dist="norm",plot=TRUE,nbin=c(20,20,10),rng=1024)
res=BANFF2(net=simdata$net,test.stat=simdata$testcov,niter=100,na.action="NN")
classLabelEst=SummaryClassLabel(simdata$net,simdata$testcov,res$zTrack,
method="MajorVote",nburn=10)
print(table(classLabelEst))
```

---

- \( \text{rhoLowB} \) \( \text{default} = c(0,0,0,0) \)
- \( \text{rhoUpB} \) \( \text{default} = c(1.5,1.5,1.5,1.5) \)
- \( \text{piLowB} \) \( \text{default} = c(0,0,0) \)
- \( \text{piUpB} \) \( \text{default} = c(1,1,1) \)
- \( \text{niter} \) \( \text{default} = 1 \)
- \( \text{replaceInf} \) \( \text{default} = -99999 \)
- \( \text{DMHplot} \) \( \text{default} = \text{FALSE} \)
**class.label**

*Gene class labels dataset*

**Description**

The simulated class labels for each gene node, by merged community detection algorithm.

**Usage**

```r
data(class.label)
```

**Format**

A vector of length=#genes, the value is 1, 2, 3 as down-regulated/ null/ up-regulated class.

**Details**

Based on the simulated network structure. We firstly apply the fast community detection algorithm and then gradually merge the communities based on their pair-wised between edge counts, until finally get three classes. The largest is assigned null, and then the upper/ down regulated are randomly chosen.

---

**DPM.HODC**

*Hierarchical ordered density clustering (HODC) Algorithm with input generated by DPdensity*

**Description**

Hierarchical ordered density clustering (HODC) Algorithm with input generated by DPdensity

**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(1e+05, 1), psiinv2 = solve(diag(0.5, 1)), nu1 = 4, nu2 = 4, tau1 = 1, tau2 = 100))
```

**Arguments**

- `v` number of iterations set for DPM fitting by "DPdensity"
- `pvalue` a vector of p-values obtained from large scale statistical hypothesis testing
- `DPM.mcmc` list
- `DPM.prior` list
Details

Without the information of networking, we can have an approximation to the marginal density by DPM model fitting on \( r \). Suppose the number of finite mixture normals is equal to \( L_0+L_1 \), which means the number of classes we have, we apply HODC algorithm in partitioning the \( SL_0 \) and \( SL_1 \) components into two classes. For this function, the input is generated by Mclust.

Value

A list of HODC algorithm returned parameters.

- **mean**: the means of each of two clusters for every DPM fitting by "DPdensity"
  - \( \mu_0 \): the means of the cluster with smaller mean
  - \( \mu_1 \): the means of the cluster with larger mean
- **variance**: the variance of each of two clusters for every DPM fitting by "DPdensity"
  - \( \sigma^2_0 \): the variances of the cluster with smaller mean
  - \( \sigma^2_1 \): the variances of the cluster with larger mean
- **probability**: the probability of each of two clusters for every DPM fitting by "DPdensity"
  - \( \pi_0 \): the probabilities of the cluster with smaller mean
  - \( \pi_1 \): the probabilities of the cluster with larger mean
- **classification**: The classification corresponding to each cluster for every DPM fitting by "DPdensity".

Examples

```r
## Not run:
### 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)
dpdensityHODC=DPM.HODC(v=5,pvalue)

## End(Not run)
```

Description

Hierarchical ordered density clustering (HODC) Algorithm with input generated by Mclust.

Usage

```r
EM.HODC(pvalue)
```
missloc

Arguments

- *pvalue*  
  a vector of p-values obtained from large scale statistical hypothesis testing

Details

Without the information of networking, we can have an approximation to the marginal density by DPM model fitting on \( r \). Suppose the number of finite mixture normals is equal to \( L_0 + L_1 \), which means the number of classes we have, we apply HODC algorithm in partitioning the \( SL_0 \) and \( SL_1 \)s components into two classes. For this function, the input is generated by Mclust

Value

- a list of HODC algorithm returned parameters.
- **mean**  
  the mean of each of two clusters
- **variance**  
  the variance of each of two clusters
- **pro**  
  the probability of each of two clusters
- **classification**  
  The classification corresponding to each cluster

Examples

```r
## Not run:
rstat=c(rnorm(50,mean=1),rnorm(50,mean=2),rnorm(100,mean=4),rnorm(100,mean=8))
pvalue=pnorm(-rstat)
mclustHODC=EM.HODC(pvalue)
## End(Not run)
```

missloc  
*Gene missing location dataset*

Description

The simulated missing locations

Usage

```r
data(missloc)
```

Format

- A vector of length=157, containing the missing location, resulting in a 20% missingness in the simulation study.

Details

The missing location are randomly selected among ‘non-hub’ genes for the simulated network with 776 gene nodes.
**Networks.Fast**

**Description**

The network used in simulation studies extracted from real data network, represented by adjacency matrix 0/1.

**Usage**

```r
data(net)
```

**Format**

A 0/1 adjacency matrix.

**Details**

In the simulation studies, the network is a subnetwork extracted based on real network, used in simulation studies. It contains 776 gene nodes.

---

**Networks.Fast**

**Fast Algorithm for Bayesian Network Discovery**

**Description**

Fast Algorithm for Bayesian Network Discovery

**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(1e+05, 1), psiinv2 = solve(diag(0.5, 1)), nu1 = 4, nu2 = 4, tau1
  = 1, tau2 = 100), DPparallel = 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` a n by n network configuration, n is the length of pvalue
- `iter` number of iterations. The default is 5000
- `nburns` number of burn-in. The default is 2000
- `algorithms` vector, either "EM" or "DPM"
\( v \) number of iterations set for DPM fitting by "DPdensity". \( v \) is only valid when you choose initials as "DPdensity"

DPM.mcmc a list
DPM.prior a list
DPparallel logical.
n.cores number of cores
piall a vector of selections of \( \pi_0 \). The default vector is 0.75, 0.8, 0.85, 0.9. The selections of \( \pi_0 \) should be placed in sequence, from smaller to larger.
rhoall a vector of selections of \( \rho_0 \) and \( \rho_1 \). The default vector is 1, 2, 5, 10, 15. The selections of \( \rho_0 \) and \( \rho_1 \) should be placed in sequence, from smaller to larger.
n.showsteps number, default=10
showlikelihood logical, default=FALSE
likelihood.frequency number, default=100

Details

This generic function fits a Bayesian Nonparametric Mixture Model for gene selection incorporating network information (Zhao et al., 2014):

- \( r_{il} \sim N(\mu_g, \sigma_g) \),
- \( g_{i \mid z_i=k} \sim \text{Discrete}(a_k, q_k) \),
- \( \theta \sim G_0k \), for \( g \in a_k \),
- \( q_k \sim \text{Dirichlet}(\tau_k 1_{L_k/L_k}) \),
- \( \theta_{g} = \theta_{g \mid g \in a_0 \text{ and } a_1} \)
- \( \theta_{g} = (\mu_g, \sigma_g) \)

where we define

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

**Assumption** We have an assumption that "selected" gene or image pixel should have larger statistics comparing to "unselected" ones without the loss of generality. In this regard, we set the restriction \( \mu_g < \mu_{g+1} \) for \( g = -L_0+1, -L_0+2, ..., L_1 \).

For this function, The NET-DPM-3, considered as Fast function is applied, and more details about the algorithm can be referred from Appendix B.3 of Zhao et al., 2014.

Value

a list containing Bayesian Statistics information of the distribution of \( z_i \)

trace a n by (iter-nburns) matrix (n is the length of elements and iter-nburns is the length of the saved chain), showing the evolution of the chain for each element

mean the mean of the distribution for each element
**median** the median of the distribution for each element

**var** the variance of the distribution for each element

**quantile** the quantiles of the distribution for each element

### Examples

```r
### Not run:
## Example1. For a 10X10 image with 5x5 signal for example
## Creating the network of 10X10 image
library(igraph)
library(BayesNetDiscovery)
g <- graph.lattice(length=10, dim=2)
##the input of argument `code(net)
net <- as(get.adjacency(g, attr=NULL), "matrix")
## Assign the signal elements with signal intenstion 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)
totalR <- Networks.Fast(pvalue, net, iter = 5000, piall = c(0.8, 0.85, 0.9, 0.95),
                        rhoall = c(0.5, 1, 5, 10, 15))
```

### Example2. Gene Network discovery

```r
## Generating Scale free Gene Network
library(igraph)
library(BayesNetDiscovery)
g <- barabasi.game(50, power = 1, zero.appeal = 1.5, directed = F)
tkplot(g, layout = layout.kamada.kawai)
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(5, 4)
  sigma = c(1, 0.5)
```
Networks.STD

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]<-rnorm(1)
  }
}
pvalue=pnorm(-testcov)
total1=Networks.Fast(pvalue, net, iter=5000, nburns=2000, piall=c(0.8, 0.85, 0.9, 0.95), rhoall=c(1, 2, 5, 10, 15))

## End(Not run)

Networks.STD Standard Algorithm for Bayesian Network Discovery

Description

Standard Algorithm for Bayesian Network Discovery

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**: a n by n network configuration, n is the length of pvalue
- **iter**: number of iterations. The default is 5000
- **nburns**: number of burn-in. The default is 2000
- **piall**: a vector of selections of pi0. The default vector is 0.75, 0.8, 0.85, 0.9. The selections of pi0 should be placed in sequence, from smaller to larger.
- **rhoall**: a vector of selections of rho0 and rho1. The default vector is 0.5, 1, 5, 10, 15. The selections of rho0 and rho1 should be placed in sequence, from smaller to larger.
- **status**: default=FALSE
- **fit**: NULL
- **show.steps**: number default=1
- **showlikelihood**: logical default=FALSE
- **likelihood.frequency**: number default=100
Details

This generic function fits a Bayesian Nonparametric Mixture Model for gene 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 \( g \in a_k \),
- \( q_k \sim \text{Dirichlet}(\tau_k 1_{L_k/L_k}) \),
- \( \theta = \theta_{g \in a_0 \text{ and } a_1} \),
- \( \theta_{g} = (\mu_{g}, \sigma_{g}) \)

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, \( \Pr(g_i=L_0+2) = q_{-L_0+2} \).

**Assumption** We have an assumption that "selected" gene or image pixel should have larger statistics comparing to "unselected" ones without the loss of generality. In this regard, we set the restriction \( \mu_{g} < \mu_{g+1} \) for \( g = -L_0+1, -L_0+2,...,L_1 \).

For this function, The NET-DPM-1, considered as standard function is applied , and more details about the algorithm can be referred from Appendix B.1 of Zhao et al., 2014.

Value

The trace of \( g_i \) showing the evolution of the Monte Carlo Markov Chain

References


Examples

```r
# Not run:
library(igraph)
library(BayesNetDiscovery)
g <- graph.lattice(length=10,dim=2)
net <- as(get.adjacency(g,attr=NULL),"matrix")
## Assign the signal elements with signal intenstion 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
```
SimulatedDataGenerator

Simulation dataset generator

Description

Function used to generating simulated dataset. See details in simulation studies

Usage

SimulatedDataGenerator(net= NULL, nnode = NULL, maxpernull = 0.7, class.label = NULL, missloc = NULL, missing = c(FALSE, TRUE), missrate = 0.1, nonmiss.hub.maxedges = 7, maxnsteps.merge.communities = 1000, dist = c("norm", "gamma", "lognorm"), plot = c(TRUE, FALSE), nbin = c(20, 20, 20), rng = 1024)

Arguments

net
The adjacent matrix with 0/1 indicating "connected" or "not directly connected. If not given, generate a scale-free graphs according to the Barabasi-Albert model by applying the BA algorithm in igraph package.

nnode
Integer. Total number of gene nodes in network.

maxpernull
float. Max percent of null genes in the network. Used when class.label is not given and needs to be generated during merging process. Default=0.7

class.label
Vector of length(total number of nodes), giving the class indicators: -1, 0, 1 to each of the gene node. If not given, class labels are defined based on fast.greedy community detection algorithm, then merged to three sequentially based on the number of between-community-edges. Highly connected communities are merged to one first. Then the largest communities are assigned class indicator 0 as null genes. The up/down regulated class are assigned randomly.
SimulatedDataGenerator

missloc  Vector. Default NULL. If given, it is the location of the test statistics that is not been observed.

missing  Logical. Default FALSE. If TRUE, the missing location are generated based on missing rate.

missrate  A number between (0,1). The missing rate defined as the proportion of gene nodes without observed test statistics. Not recommend over 20% based on biological knowledge.

nonmiss.hub.maxedges  Integer. Based on biological knowledge, hub genes (with higher number of neighboring edges) are less likely to be missing gene nodes. Thus it is the cutoff value where only genes with less than the nonmiss.hub.maxedges neighbors can be assigned as missing genes. Default=7

maxnsteps.merge.communties  Integer. The maximum number of steps used for merging the small communities. In order to be 3, defual=1000.

dist  Char. The distribution of DE genes, can be one of the following: c("norm","gamma","lognorm"). See details in simulation design table.

plot  Logical. Defaul=TRUE: whether to plot the histogram of test statistics being generated or not.

nbin  Vector of length 3. Default=c(5,20,5). The number of bins used for ploting the histogram for each of the class.

rng  Random seed Defaul=1024

Details

The function used for simulating test statistics:

- network is given or generated by Barabasi-Albert algorithm in igraph package.
- class indicators is given or generated based on fast.greedy community detection algorithm.
- test statistics, currently support three simulation scenario: c("norm","gamma","lognorm")

Value

A list:

testcov  test statistics, missing observations are coded as NA if any

testcov.fullobs  test statistics when all the observations are fully observed

class.label  z values for each gene, class indicators

net  simulated network, binary adjacency matrix 1/0 connected or not
SummaryClassLabel

Examples

```r
## Not run:
## The simulation settings based on real gene network. (takes time)
data(net)
data(class.label)
data(missloc)
simdata=SimulatedDataGenerator(net=net,class.label=class.label,missloc=missloc,
dist="norm",plot=TRUE,nbin=c(20,20,20),rng=1024)
str(simdata)
## A toy example
simdata=SimulatedDataGenerator(nnode=100,missing=TRUE,missrate=0.1,dist="norm",
plot=TRUE,nbin=c(20,20,10),rng=1024)
str(simdata)
## End(Not run)
```

SummaryClassLabel Local FDR method for summarizing class labels

Description

The finalized class label can be either by method "Major vote", or "under local fdr control".

Usage

```r
SummaryClassLabel(net,test.stat,zTrack,method=c("MajorVote","LocfdrControl"),
plot=FALSE,nburn=50,nskip=2,locfdr.alpha=0.2)
```

Arguments

- `net` The adjacent matrix with 0/1 indicating "connected" or "not directly connected
- `test.stat` The observed test statistics. Missing values are represented as NAs.
- `zTrack` The trace for z. It is a num_of_genes by num_of_iterations matrix.
- `method` A char. Either "MajorVote" or "LocfdrControl".
- `plot` Logical. Default=FALSE, means the plot should be drawn.
- `nburn` Default=50.
- `nskip` Default=2
- `locfdr.alpha` Default=0.2

Value

A vector, where each value is the summarized class label in c(1,2,3) representing "down-regulate", "null", and "up-regulate".
Examples

```r
## Not run:
data(net)
data(test.stat)
res=FeatureClassFinder(net,test.stat,niter=100,na.action="NN")
classLabelEst=SummaryClassLabel(net,test.stat,res$zTrack,method="MajorVote")
print(table(classLabelEst))

## End(Not run)
```

---

test.stat  

**Gene test statistics dataset**

Description

The simulated test statistics on each gene node. With 10% randomly selected as missing.

Usage

```r
data(test.stat)
```

Format

A vector of length=#genes, the value is the observed test statistics for each gene.

Details

We first simulate the underlying true feature labels by 25% down-regulate, 50% null and 25% up-regulate. The pre-defined distribution for each regulation group is normal: the down-regulate genes are distributed as a normal with mean=-0.5 and sd=0.2, the null genes are distributed as a normal with mean=0 and sd=0.2, the up-regulated genes are distributed as a normal with mean=0.5 and sd=0.2. For the second step, we randomly knock out 10% genes as missing nodes.
Index

*Topic datasets
  class.label, 5
  missloc, 7
  net, 8
  test.stat, 16

BANFF2, 2

class.label, 5

DPM.HODC, 5

EM.HODC, 6

missloc, 7

net, 8

Networks.Fast, 8

Networks.STD, 11

SimulatedDataGenerator, 13

SummaryClassLabel, 15

test.stat, 16