Package ‘G1DBN’

February 19, 2015

Version 3.1.1

Date 2012-05-23

Title A package performing Dynamic Bayesian Network inference.

Author Sophie Lebre <sophie.lebre@icube.unistra.fr>, original
version 1.0 by Sophie Lebre

<sophie.lebre@icube.unistra.fr>, contribution of Julien
Chiquet <julien.chiquet@genopole.cnrs.fr> to version 2.0

Maintainer Sophie Lebre <sophie.lebre@icube.unistra.fr>

Depends R (>= 2.11.1), MASS, igraph

Description G1DBN performs DBN inference using 1st order conditional
dependencies.

License GPL (>= 2)

Repository CRAN

Date/Publication 2013-09-05 19:27:45

NeedsCompilation no

R topics documented:

<table>
<thead>
<tr>
<th>R topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>arth800line</td>
<td>2</td>
</tr>
<tr>
<td>BuildEdges</td>
<td>3</td>
</tr>
<tr>
<td>BuildNetwork</td>
<td>5</td>
</tr>
<tr>
<td>DBNScoreStep1</td>
<td>8</td>
</tr>
<tr>
<td>DBNScoreStep2</td>
<td>11</td>
</tr>
<tr>
<td>PRCurve</td>
<td>14</td>
</tr>
<tr>
<td>ROCurve</td>
<td>16</td>
</tr>
<tr>
<td>SimulGeneExpressionAR1</td>
<td>18</td>
</tr>
<tr>
<td>SimulNetworkAdjMatrix</td>
<td>20</td>
</tr>
</tbody>
</table>

Index 22
Description

This data set describes the temporal log2 transformed expression of 800 genes of A. thaliana during the diurnal cycle. The data are in line, that is 2 repeated measurements time series are displayed one after the other, separated by a 'NA' value. The 800 genes are a subset of the data presented in Smith et al. (2004) selected for periodicity according to the method implemented in the R package GeneCycle (http://strimmerlab.org/software/genecycle/).

Usage

data(arth800line)

Format

matrix with 800 columns (=genes) and 23 rows (rows 1 to 11 contain the first measurement time series, row 12 contain 'NA' values and rows 13 to 23 contain the second experiment time series).

Author(s)


Source

The microarray experiments were performed in the laboratory of S. Smith (Edinburgh). The data are available from the NASCArrays database (http://affymetrix.arabidopsis.info/ under experiment reference number NASCARRAYS-60).

References


Examples

```r
## load G1DBN library
library(G1DBN)

## load data set
data(arth800line)
ids<-c(60, 141, 260, 333, 365, 424, 441, 512, 521, 578, 789, 799)

## plot first ten time series
plot(1:23, arth800line[, ids], type="l", ylim=c(2,12), xlab="Time",
```
BuildEdges

Description

Given a score matrix, this function builds the list of the edges of the associated network. The edges are ordered according to their scores. The score matrix has been computed from a network inference algorithm (e.g. DBNScoreStep1 or DBNScoreStep2, Shrinkage, Lasso, ...). An optional threshold can be specified, as well as a maximal number of edges.

Usage

BuildEdges(score, threshold=1, nb=NULL, targetNames=NULL, predNames=NULL, prec=3, dec=FALSE)

Arguments

- **score**: matrix with \( r \) rows (=target genes) and \( d \) columns (=predictor genes) containing the scores resulting from an estimation procedure (e.g. DBNScoreStep1 or DBNScoreStep2, Shrinkage, Lasso, ...).
- **threshold**: An optional real setting the maximal value for edge selection, default=1.
- **nb**: An optional integer setting the maximal number of selected edges, default=NULL.
- **targetNames**: An optional array \( (r) \) giving a list of names for the target genes, default=NULL.
- **predNames**: An optional array \( (d) \) giving a list of names for the predictor genes, default=NULL.
- **prec**: An optional integer setting the number of decimal places for score display, default=3.
- **dec**: boolean, FALSE if the smallest score points out the most significant edge, default=FALSE.

Value

A matrix containing a list of edges ordered according to the score (First column: predictor, second column: target, third column: corresponding score). Predictors and targets are referred to through the names given by targetNames or predNames when specified.
Author(s)

Chiquet Julien (http://stat.genopole.cnrs.fr/~jchiquet).

See Also

DBNScoreStep1, DBNScoreStep2, BuildNetwork

Examples

library(G1DBN)

# SIMULATING THE NETWORK

# number of genes
p <- 10
# the network - adjacency Matrix
MyNet <- SimulNetworkAdjMatrix(p, 0.05, c(-1.5, -0.5, 0.5, 1.5))
MyNet

# SIMULATING THE TIME SERIES EXPERIMENTS

# number of time points
n <- 20
# initializing the B vector
B <- runif(p, -1, 1)
# initializing the variance of the noise
sigmaEps <- runif(p, 0.1, 0.5)
# initializing the process Xt
X0 <- B + rnorm(p, 0, sigmaEps * 10)
# the times series process
Xn <- SimulGeneExpressionAR1(MyNet$A, B, X0, sigmaEps, n)

# NETWORK INFERENCE WITH G1DBN

# STEP 1 - The first step score matrix
S1 <- DBNScoreStep1(Xn, method = 'ls')

# Building the edges of the network inferred after Step1
alpha1 <- 0.5
G1 <- BuildEdges(S1$S1ls, threshold = alpha1, dec = FALSE)

# STEP 2 - The second step score matrix
S2 <- DBNScoreStep2(S1$S1ls, Xn, method = 'ls', alpha1)

# Building the edges of the network inferred after Step2
alpha2 <- 0.05
G2 <- BuildEdges(S2, threshold = alpha2, dec = FALSE)

G2
## BuildNetwork

**Network object creation**

**Description**

Given a list of scored edges and a REQUIRED vector of labels (e.g., 1:p), this function builds an object "Network". This is useful for exporting a network described by a list of edges to a network described by an adjacency matrix.

**Usage**

```r
BuildNetwork(Edges, Labels, nonedges.val = NA)
```

**Arguments**

- **Edges**: a \( n \times 3 \) matrix (each line contains a couple of vertices plus the associated score)
- **Labels**: a vector of labels for the vertices
- **nonedges.val**: optional. Value attributed to the not existing edges in the generated score matrix out$Score, default=NA

**Value**

A list that contains:
- `out$Vertices$Num` the number of vertices,
- `out$Vertices$Labels` a vector of labels of the vertices,
- `out$Vertices$Connected` a vector of the connected vertices,
- `out$Edges$Prop` the proportion of edges,
- `out$Edges$Num` the number of edges,
- The graph of the network (out$AdjMatrix an adjacency matrix and out$Edges$List a list of edges) and `out$Score` a score matrix.

**Author(s)**

- Chiquet Julien ([http://stat.genopole.cnrs.fr/~jchiquet](http://stat.genopole.cnrs.fr/~jchiquet)).

**See Also**

- BuildEdges
Examples

```r
library(G1DBN)

## == SIMULATING THE NETWORK

## number of genes
p <- 10
## number of time points
n <- 20
## proportion of genes
geneProp <- 0.05
## the network - adjacency Matrix
MyNet <- SimulNetworkAdjMatrix(p,geneProp,c(-1.5,-0.5,0.5,1.5))

cat("\n===============\nSIMULATION\n")

cat("\n== SIMULATING THE TIME SERIES EXPERIMENTS
")

## Autoregressive model

## initializing the b vector
b <- runif(p,-1,1)
## initializing the variance of the noise
sigmaEps <- runif(p,0.1,0.5)
## initializing the process X_t
X0 <- B + rnorm(p,0,sigmaEps*10)
## the times series process
Xn <- SimulGeneExpressionAR1(MyNet$A,B,X0,sigmaEps,n)

## NETWORK INFEERENCE WITH G1DBN

cat("\n== NETWORK INFERENCE\n")
cat("Using a Dynamic Bayesian Network model\n")

## STEP 1

## step 1

cat("STEP 1...\n")
S1 <- DBNScoreStep1(Xn, method='ls')

## STEP 2

## step 2

cat("STEP 2...\n")
```
BuildNetwork

alpha1=0.5
S2 <- DBNScoreStep2(S1$S1s, data=Xn, method='ls', alpha1=alpha1)

# POST TREATMENTS

# building the inferred Graph
G1 <- BuildEdges(S1$S1s, threshold=alpha1, dec=FALSE)

# encoding as the adjacency matrix graph
Step1InferredNet <- BuildNetwork(G1,1:p)
Step1InferredNet

# Step 2
alpha2=0.05
G2 <- BuildEdges(S2, threshold=alpha2, dec=FALSE)
Step2InferredNet <- BuildNetwork(G2,1:p)

# Not run:
cat("\n=")
cat("SUMMARY\n")
cat("Plotting the results...\n")
split.screen(c(1,3))

# The Original graph and data
# --------------------------
# set the edges list of the simulated network
G0 <- BuildEdges(MyNet$AdjMatrix, threshold=0.9, dec=TRUE)

# Nodes coordinates are calculated according to the global structure of the network
all_parents=c(G0[,1], G1[,1], G2[,1])
all_targets=c(G0[,2], G1[,2], G2[,2])
posEdgesG0=1:dim(G0)[1]
posEdgesG1=(dim(G0)[1]+1):(dim(G0)[1]+dim(G1)[1])
posEdgesG2=(dim(G0)[1]+dim(G1)[1]+1):length(all_parents)

# Global network with all the edges
netAll =
graph.edgelist(cbind(as.character(all_parents), as.character(all_targets)))

# Nodes coordinates
nodeCoord=layout.fruchterman.reingold(netAll)

# after Step 1
screen(1)
# set the edges list
netG1 = graph.edgelist(cbind(as.character(G1[,1]), as.character(G1[,2])))
First order dependence graph $G(1)$ inference

Description

Given a time series dataset for $p$ genes, this function infers a 1st order dependence score matrix $S_1 \ (p \times p)$ which contains the score of each edge of a Dynamic Bayesian Network (DAG $G(1)$) describing first order dependencies between successives variables. The smallest score points out the most significant edge for the 1st order dependence DAG $G(1)$. The sets of both predictor and target genes can be reduced to different subsets of the $p$ genes. DBNScoreStep1 is the first step of the estimation procedure described in the references. See function DBNScoreStep2 to perform the second step selection and infer a score matrix describing full order dependencies.

Usage

```r
DBNScoreStep1(data, method='ls', predPosition=NULL, targetPosition=NULL)
```
Arguments

- **data**: A matrix with $n$ rows (=time points) and $p$ columns (=genes) containing the gene expression time series.
- **method**: Currently M estimation with either LS, Tukey bisquare or Huber estimator: c('ls','tukey','huber'), default='ls'.
- **predPosition**: To be specified to reduce the set of possible predictor genes to a subset of $d < p$ genes: an array included in [1, p] defining the position of the d predictor genes in the data matrix ($n \times p$), default=NULL.
- **targetPosition**: To be specified to reduce the set of possible target genes to a subset of $r < p$ genes: an array included in [1, p] defining the position of the r target genes in the data matrix ($n \times p$), default=NULL.

Value

A list with out$S1ls a matrix with min($r, p$) rows (=target genes) and min($d, p$) columns (=predictor genes) containing the scores $S1$ obtained with least square estimator, out$S1huber a matrix containing scores $S1$ obtained with Huber estimator, out$S1tukey a matrix containing scores $S1$ obtained with Tukey bisquare (or biweight) estimator.(out$S1ls[i,j] is the score for the edge $j \leftarrow i$ pointing out from predictor $j$ toward target $i$.)

Note

For a large number of target genes, it is of interest to parallel run the procedure DBNScoreStep1 for each target gene by running $p$ the following jobs for $i = 1 \ldots p$, outi <- DBNScoreStep1(data, target=i).

Author(s)


References

Lebre, S. 2009. Inferring dynamic bayesian network with low order independencies, Statistical Applications in Genetics and Molecular Biology, 2009: Vol. 8: Iss. 1, Article 9.

See Also

DBNScoreStep2, BuildEdges, PRcurve.

Examples

```r
## load G1DBN Library
library(G1DBN)

data(arth8001line)
data<-as.matrix(arth8001line)
id<-c(60, 141, 260, 333, 365, 424, 441, 512, 521, 578, 789, 799)
```
names<-c("carbohydrate/sugar transporter","ATGPX2","putative integral membrane prot", "AT3G05900", "At3g27350", "At1g16720", "ATISA3/ISA3", "AT4G32190", "catalase", "plasma membrane intrinsic prot", "At4g16146", "DPE2")

## compute score S1
out<-DBNScoreStep1(data,method='ls', targetPosition=id,predPosition=id)
round(out$s1ls,2)

## Threshold for the selection of the edges after Step 1
alpha1=0.5
## Build the edges with id as label
edgesG1id<-BuildEdges(score=out$s1ls,threshold=alpha1,
targetNames=id,predNames=id,prec=6)
## Build the edges with names as label
edgesG1names<-BuildEdges(score=out$s1ls,threshold=alpha1,
targetNames=names,predNames=names,prec=6)
edgesG1id[1:15,]
edgesG1names[1:15,]

## compute score S2 from S1
S2<-DBNScoreStep2(out$s1ls,data,method='ls',alpha1=alpha1,
predPosition=id,targetPosition=id)
S2

## Threshold for the selection of the edges after Step 2
alpha2=0.05
## Build the edges with id as label
edgesG2id<-BuildEdges(score=S2,threshold=alpha2,
targetNames=id,predNames=id,prec=6)
## Build the edges with names as label
edgesG2names<-BuildEdges(score=S2,threshold=alpha2,
targetNames=names,predNames=names,prec=6)
edgesG2id
edgesG2names

## As the number of genes is reduced to 10 here, this results slightly differ
## from the results obtained in the paper (Lebre, 2009) cited in References.

## -------------------------------
## PLOTTING THE RESULTS...
## -------------------------------
## Not run:
## The Inferred Nets
## -------------------------------

## Nodes coordinates are calculated according to the global structure of the graph
all_parents=c(edgesG1id[,1], edgesG2id[,1])
all_targets=c(edgesG1id[,2], edgesG2id[,2])
Given a time series dataset for \( p \) genes, a 1st order dependence score matrix \( S_1 \) (obtained with function \( \text{DBNScoreStep1} \)) and a threshold \( \alpha_1 \) for edge selection in matrix \( S_1 \), this function infers the score of each edge of a Dynamic Bayesian Network (DAG \( G \)) describing full order dependencies between successive variables. This is the second step of the inference procedure described.
in the references. 1st step DBNScoreStep1 allows to reduce the number of potential edges, DBNScoreStep2 performs the last step selection. The smallest score points out the most significant edge.

Usage

DBNScoreStep2(S1, data, method='ls', alpha1, predPosition=NULL, targetPosition=NULL)

Arguments

S1 a matrix with \( r \) rows (=target genes) and \( d \) columns (=predictor genes) containing score \( S1 \) (maximal p-value) obtained with function DBNScoreStep1.

data a matrix with \( n \) rows (=time points) and \( p \) columns (=genes) containing the gene expression time series.

method one of 'ls' (default), 'huber', 'tukey'. This specifies the regression method.

alpha1 Threshold for edge selection in the 1st order dependence score matrix \( S1 \). Edges having a score greater than \( \alpha1 \) are pruned and quoted 'NA' is the resulting score matrix \( S2 \).

predPosition To be specified if the number \( d \) of predictor genes in score matrix \( S1 \) is lower than the number \( p \) of genes in the data: an array included in \([1, p]\) defining the position of the \( d \) predictor genes in the data matrix \((n \times p)\), default=NULL.

targetPosition To be specified if the number \( r \) of target genes in score matrix \( S1 \) is lower than the number \( p \) of genes in the data: an array included in \([1, p]\) defining the position of the \( r \) target genes in the data matrix \((n \times p)\), default=NULL.

Value

A matrix \((r \times d)\) containing the scores \( S2 \) obtained after the second step inference with the chosen M estimator. The score of the edges pruned after the first step inference is 'NA'.

Author(s)


Chiquet Julien (http://stat.genopole.cnrs.fr/~jchiquet).

References

Lebre, S. 2009. Inferring dynamic bayesian network with low order independencies, Statistical Applications in Genetics and Molecular Biology, 2009: Vol. 8: Iss. 1, Article 9.

See Also

DBNScoreStep1, BuildEdges.
Examples

```r
## load G1DBN Library
library(G1DBN)

data(arth800line)
data<-as.matrix(arth800line)
id<-c(60, 141, 260, 333, 365, 424, 441, 512, 521, 578, 789, 799)
names<-c("carbohydrate/sugar transporter","ATGFX2","putative integral membrane prot",
"AT3G05900", "At3g27350", "At1g16720","ATISA3/ISA3","AT4G32190", "catalase", "plasma membrane intrinsic prot", "At4g16146", "DPE2")

## compute score S1
out<-DBNScoreStep1(data, method='ls', targetPosition=id,predPosition=id)
round(out$S1ls,2)

## Threshold for the selection of the edges after Step 1
alpha1=0.5
## Build the edges with id as label
edgesG1id<-BuildEdges(score=out$S1ls,threshold=alpha1,
targetNames=id,predNames=id,prec=6)
## Build the edges with names as label
edgesG1names<-BuildEdges(score=out$S1ls,threshold=alpha1,
targetNames=names,predNames=names,prec=6)
edgesG1id[1:15]
edgesG1names[1:15]

## compute score S2 from S1
S2<-DBNScoreStep2(out$S1ls,data,method='ls',alpha1=alpha1,
predPosition=id,targetPosition=id)
S2

## Threshold for the selection of the edges after Step 2
alpha2=0.05
## Build the edges with id as label
edgesG2id<-BuildEdges(score=S2,threshold=alpha2,
targetNames=id,predNames=id,prec=6)
## Build the edges with names as label
edgesG2names<-BuildEdges(score=S2,threshold=alpha2,
targetNames=names,predNames=names,prec=6)
edgesG2id
edgesG2names

## As the number of genes is reduced to 10 here, this results slightly differ
## from the results obtained in the paper (Lebre, 2009) cited in References.

## -----------------------------------------------
## PLOTTING THE RESULTS...
## Not run:
### The Inferred Nets

```r
# Nodes coordinates are calculated according to the global structure of the graph
all_parents = c(edgesG1id[,1], edgesG2id[,1])
all_targets = c(edgesG1id[,2], edgesG2id[,2])
posEdgesG1 = 1:dim(edgesG1id)[1]
PosEdgesG2 = (dim(edgesG1id)[1]+1):length(all_targets)

# Global network with all the edges
netAll = graph.edgelist(cbind(as.character(all_parents), as.character(all_targets)))

# Nodes coordinates
nodecoord = layout.fruchterman.reingold(netAll)
```

```
split.screen(c(1,2))

# after Step 1
screen(1)
# set the edges list
netG1 = graph.edgelist(cbind(as.character(edgesG1id[,1]), as.character(edgesG1id[,2])))
# set the object for plotting the network with global coordinates of all nodes
G1toPlot = delete.edges(netAll, E(netAll)[posEdgesG2])
# plot the network
plot(G1toPlot, layout=nodeCoord, vertex.label =
    get.vertex.attribute(G1toPlot, name="name"), edge.arrow.size = 0.2,
    main="GIDBN Inferred network:
    \n    Step 1")

# after Step 2
screen(2)
# set the edges list
netG2 = graph.edgelist(cbind(as.character(edgesG2id[,1]), as.character(edgesG2id[,2])))
# set the object for plotting the network with global coordinates of all nodes
G2toPlot = delete.edges(netAll, E(netAll)[posEdgesG1])
# plot the network
plot(G2toPlot, layout=nodeCoord, vertex.label =
    get.vertex.attribute(G2toPlot, name="name"), edge.arrow.size = 0.2,
    main="GIDBN Inferred network:
    \n    Step 2")

close.screen(all = TRUE)
```

## End(Not run)
Description
Given a score matrix and a validation matrix, this function allows to compute the corresponding Precision-Recall (PR) curve by returning a list with the respective $x$ coordinates (recall) and $y$ coordinates (precision) of the PR curve. The recall is equal to the sensitivity, that is the number of true positive out of the number of true edges to be detected. The precision is the Positive Predictive Value, that is the number of true positive edges out of the number of selected edges. The score matrix has been computed from a network inference algorithm (e.g. DBNScoreStep1 or DBNScoreStep2, Shrinkage, Lasso, ...).

Usage
PRcurve(score, validMat, dec=FALSE)

Arguments
score matrix with $r$ rows (=target genes) and $d$ columns (=predictor genes) containing the scores resulting from an estimation procedure (e.g. DBNScoreStep1 or DBNScoreStep2, Shrinkage, Lasso, ...).
validMat An optional matrix specifying the validated edges (1 if an edge is validated, 0 otherwise).
dec boolean, FALSE if the smallest score points out the most significant edge, default=FALSE.

Value
A list with out$recall and out$precision containing respectively the recall values ($x$ coordinates of the PR curve) and the precision values ($y$ coordinates).

Author(s)
Chiquet Julien (http://stat.genopole.cnrs.fr/~jchiquet).

See Also
DBNScoreStep1, DBNScoreStep2, ROCcurve.

Examples
library(G1DBN)
## generate the validation matrix
## number of genes
p <- 20
## the network - adjacency Matrix
MyNet <- SimulNetworkAdjMatrix(p, 0.05, c(-1, 0.5, 1))

## generate the time series
## initializing the B vector
B <- runif(p, -1, 1)
## initializing the variance of the noise
sigmaEps <- runif(p,0.05,0.5)
## initializing the process Xt
X0 <- B + rnorm(p,0,sigmaEps*10)
## number of time points
n <- 20

## the AR(1) times series process
Xn <- Simul GeneExpressionAR1(MyNet$A,B,X0,sigmaEps,n)

## compute score S1
out<DBN Score Step 1(Xn)
pr1<PRCurve(score=out$S1s,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

## compute score S2 from S1
## depending on the generated data, the threshold alpha1 has to be chosen differently.
alphal<0.8
S2<DBN Score Step 2(S1=out$S1s,data=Xn,alphal=alphal)
pr2.0.8<PRCurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)
alphal<0.6
S2<DBN Score Step 2(S1=out$S1s,data=Xn,alphal=alphal)
pr2.0.6<PRCurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)
alphal<0.4
S2<DBN Score Step 2(S1=out$S1s,data=Xn,alphal=alphal)
pr2.0.4<PRCurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)
alphal<0.2
S2<DBN Score Step 2(S1=out$S1s,data=Xn,alphal=alphal)
pr2.0.2<PRCurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

plot(pr1$recall,pr1$precision,type="l",main="PR curves after both Step 1 and Step 2",
ylab="ppv",xlab="Sensitivity",lwd=2,xlim=c(0,1),ylim=c(0,1),lty=2)
lines(pr2.0.8$recall,pr2.0.8$precision, col=3,lwd=2)
lines(pr2.0.6$recall,pr2.0.6$precision, col=4,lwd=2)
lines(pr2.0.4$recall,pr2.0.4$precision, col=5,lwd=2)
lines(pr2.0.2$recall,pr2.0.2$precision, col=6,lwd=2)
lines(0:1,c(0,0),lty=3)
lines(0:1,c(1,1),lty=3)
lines(c(0,0),0:1,lty=3)
lines(c(1,1),0:1,lty=3)
leg=c("Step 1", "Step 2 (alphal=0.8)",  "Step 2 (alphal=0.6)",
  "Step 2 (alphal=0.4)", "Step 2 (alphal=0.2)")
legend(0,0.25,leg, lty=c(2,1,1,1,1), col=c(1,3,4,5,6),lwd=array(2,5))
**Description**

Given a score matrix and a validation matrix, this function allows to compute the corresponding ROC curve by returning a list with the respective $x$ and $y$ coordinates of the ROC curve. The score matrix has been computed from a network inference algorithm (e.g. DBNScoreStep1 or DBNScoreStep2, Shrinkage, Lasso, ...).

**Usage**

```r
ROCcurve(score, validMat, dec=FALSE)
```

**Arguments**

- `score`: matrix with $r$ rows (=target genes) and $d$ columns (=predictor genes) containing the scores resulting from an estimation procedure (e.g. DBNScoreStep1 or DBNScoreStep2, Shrinkage, Lasso, ...).
- `validMat`: An optional matrix specifying the validated edges (1 if an edge is validated, 0 otherwise).
- `dec`: boolean, FALSE if the smallest score points out the most significant edge, default=FALSE.

**Value**

A list with out$x$ and out$y$ contain respectively the $x$ and the $y$ coordinates of the ROC curve.

**Author(s)**

Chiquet Julien ([http://stat.genopole.cnrs.fr/~jchiquet](http://stat.genopole.cnrs.fr/~jchiquet)).

**See Also**

DBNScoreStep1, DBNScoreStep2, PRcurve.

**Examples**

```r
## generate the validation matrix
## number of genes
p <- 20
## the network - adjacency Matrix
MyNet <- SimulNetworkAdjMatrix(p, 0.05, c(-1, 0.5, 0.5, 1))

## generate the time series
## initializing the B vector
B <- runif(p,-1,1)
## initializing the variance of the noise
sigmaEps <- runif(p, 0.05, 0.5)
## initializing the process X_t
X0 <- B + rnorm(p, 0, sigmaEps*10)
## number of time points
n <- 20
```
## the AR(1) times series process

\[ X_n \leftarrow \text{SimulGeneExpressionAR1}(\text{MyNet}$A,B,X_0,\text{sigmaEps},n) \]

## compute score S1

\[ \text{out}<-\text{DBNScoreStep1}(X_n) \]

\[ \text{roc1}<-\text{ROCcurve}(\text{score=out}$S1ls,\text{validMat=abs(MyNet}$AdjMatrix)>0,\text{dec=FALSE}) \]

## compute score S2 from S1

## depending on the generated data, the threshold \( \alpha_1 \) has to be chosen differently.

\[ \alpha_1=0.8 \]

\[ S2<-\text{DBNScoreStep2}(S1=out$S1ls,\text{data=Xn,}\alpha_1=\alpha_1) \]

\[ \text{roc2}_0.8<-\text{ROCcurve}(\text{score=S2,validMat=abs(MyNet}$AdjMatrix)>0,\text{dec=FALSE}) \]

\[ \alpha_1=0.6 \]

\[ S2<-\text{DBNScoreStep2}(S1=out$S1ls,\text{data=Xn,}\alpha_1=\alpha_1) \]

\[ \text{roc2}_0.6<-\text{ROCcurve}(\text{score=S2,validMat=abs(MyNet}$AdjMatrix)>0,\text{dec=FALSE}) \]

\[ \alpha_1=0.4 \]

\[ S2<-\text{DBNScoreStep2}(S1=out$S1ls,\text{data=Xn,}\alpha_1=\alpha_1) \]

\[ \text{roc2}_0.4<-\text{ROCcurve}(\text{score=S2,validMat=abs(MyNet}$AdjMatrix)>0,\text{dec=FALSE}) \]

\[ \alpha_1=0.2 \]

\[ S2<-\text{DBNScoreStep2}(S1=out$S1ls,\text{data=Xn,}\alpha_1=\alpha_1) \]

\[ \text{roc2}_0.2<-\text{ROCcurve}(\text{score=S2,validMat=abs(MyNet}$AdjMatrix)>0,\text{dec=FALSE}) \]

\[ \text{TP=}\sum\text{abs(MyNet}$AdjMatrix)>0) \]

\[ \text{FN=p^2-TP} \]

\[ \text{plot(roc1}$x/\text{FN,roc1}$y/\text{TP,type="l", main="ROC curve after both Step1 and Step2", ylab="True Positive Rate", xlab="False Negative Rate",lwd=2,lty=2)} \]

\[ \text{lines(roc2}_0.8$x/\text{FN, roc2}_0.8$y/\text{TP, col=3,lwd=2)}} \]

\[ \text{lines(roc2}_0.6$x/\text{FN, roc2}_0.6$y/\text{TP, col=4,lwd=2)}} \]

\[ \text{lines(roc2}_0.4$x/\text{FN, roc2}_0.4$y/\text{TP, col=5,lwd=2)}} \]

\[ \text{lines(roc2}_0.2$x/\text{FN, roc2}_0.2$y/\text{TP, col=6,lwd=2)}} \]

\[ \text{lines(0:1,c(0,0),lty=3)} \]

\[ \text{lines(0:1,c(1,1),lty=3)} \]

\[ \text{lines(c(0,0),0:1,lty=3)} \]

\[ \text{lines(c(1,1),0:1,lty=3)} \]

\[ \text{leg=c("Step 1", "Step 2 (alpha=0.8)", "Step 2 (alpha=0.6)", "Step 2 (alpha=0.4)", "Step 2 (alpha=0.2)")}
\]

\[ \text{legend(0.568,0.265, leg, lty=c(2,1,1,1,1), col=c(1,3,4,5,6),lwd=array(2,5))} \]

---

**First order multivariate Auto-Regressive time series generation**
**Description**

This function generates multivariate time series according to the following first order Auto-Regressive process,

\[ X(t) = AX(t - 1) + B + \varepsilon(t), \]

where \( \varepsilon(t) \) follows a zero-centered multivariate gaussian distribution whose variance matrix \( S \) is diagonal.

**Usage**

```r
SimulGeneExpressionAR1(A, B, X0, SigmaEps, n)
```

**Arguments**

- `A`: a matrix \((p \times p)\)
- `B`: a column vector \((p \times 1)\)
- `X0`: a column vector \((p \times 1)\) containing the values of the process at time 0
- `SigmaEps`: a column vector \((p \times 1)\) containing the values of the diagonal of covariance matrix \( S \)
- `n`: the desired length of the time serie.

**Value**

A matrix, with \( n \) rows (=length) and \( p \) columns (=dimension), containing the generated time series.

**Author(s)**


Chiquet Julien (http://stat.genopole.cnrs.fr/~jchiquet).

**See Also**

`SimulNetworkAdjMatrix`

**Examples**

```r
library(G1DBN)
## number of genes
p <- 20
## the network - adjacency Matrix
MyNet <- SimulNetworkAdjMatrix(p, 0.05, c(-1, 0, 0, 1))

## initializing the B vector
B <- runif(p, 0, 0.5)
## initializing the variance of the noise
sigmaEps <- runif(p, 0.1, 0.8)
## initializing the process X
X0 <- B + rnorm(p, 0, sigmaEps * 10)
## number of time points
```
n <- 30

## the AR(1) time series process
Xn <- SimulGeneExpressionAR1(MyNet$A,B,X0,sigmaEps,n)

plot(1:n, Xn[,1],type="l", xlab="Time t", ylab="X(t) ", 
main="Simulated AR(1) time series", ylim=c(min(Xn),max(Xn))

for (i in 2:p){
  lines(1:n,Xn[,i],col=i)
}

SimulNetworkAdjMatrix  Network object generation

Description

This function builds a object "Network" by simulating a matrix of valued adjacencies from a number of vertices, a proportion of edges and the range of the uniform distribution that is used to build the adjacency matrix. An optional vector of labels may be given.

Usage

SimulNetworkAdjMatrix(Num,EdgesProp,Range,Labels=1:Num)

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num</td>
<td>number of genes</td>
</tr>
<tr>
<td>EdgesProp</td>
<td>edges proportion in the network</td>
</tr>
<tr>
<td>Range</td>
<td>vector with 4 elements specifying range values for the adjacency matrix generation (minimum negative value, maximum negative value, minimum positive value, maximum positive value)</td>
</tr>
<tr>
<td>Labels</td>
<td>an optional vector of labels for the edges</td>
</tr>
</tbody>
</table>

Value

A list that contains out$Vertices$Num the number of vertices, out$Vertices$Labels a vector of labels of the vertices, out$Vertices$Regulated a vector of the regulated vertices, out$Edges$Prop the proportion of edges, out$Edges$Num the number of edges, out$AdjMatrix an adjacency matrix (binary) and out$A a valued adjacency matrix.

Author(s)

Chiquet Julien (http://stat.genopole.cnrs.fr/~jchiquet).
SimulNetworkAdjMatrix

See Also

SimulGeneExpressionAR1, BuildEdges

Examples

library(G1DBN)
## number of genes
p <- 10
## the network - adjacency Matrix
MyNet <- SimulNetworkAdjMatrix(p, 0.05, c(-1, 0, 0, 1))
MyNet

## initializing the B vector
B <- runif(p, 0, 0.5)
## initializing the variance of the noise
sigmaEps <- runif(p, 0.1, 0.8)
## initializing the process Xt
X0 <- B + rnorm(p, 0, sigmaEps * 10)
## number of time points
n <- 20

## the AR(1) times series process
Xn <- SimulGeneExpressionAR1(MyNet$AdjMatrix, B, X0, sigmaEps, n)
Index

*Topic datagen
  BuildNetwork, 5
  SimulGeneExpressionAR1, 18
  SimulNetworkAdjMatrix, 20

*Topic datasets
  arth800line, 2

*Topic graphs
  DBNScoreStep1, 8
  DBNScoreStep2, 11

*Topic models
  DBNScoreStep1, 8
  DBNScoreStep2, 11

*Topic regression
  DBNScoreStep1, 8
  DBNScoreStep2, 11

*Topic robust
  DBNScoreStep1, 8
  DBNScoreStep2, 11

*Topic ts
  DBNScoreStep1, 8
  DBNScoreStep2, 11
  SimulGeneExpressionAR1, 18

*Topic utilities
  BuildEdges, 3
  PRcurve, 14
  ROCcurve, 16

arth800line, 2
BuildEdges, 3
BuildNetwork, 5
DBNScoreStep1, 8
DBNScoreStep2, 11
PRcurve, 14
ROCcurve, 16
SimulGeneExpressionAR1, 18
SimulNetworkAdjMatrix, 20