

Package ‘G1DBN’

April 17, 2009

Version 2.0

Date 2008-01-23

Title A package performing Dynamic Bayesian Network inference.

Author original version by Sophie Lebre <s.lebre@imperial.ac.uk>, earlier work by Sophie Lebre and Julien Chiquet <julien.chiquet@genopole.cnrs.fr>

Maintainer Mark Hoebeke <Mark.Hoebeke@jouy.inra.fr>

Depends R (>= 2.5.1), MASS, sna

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

License GPL (>= 2)

Repository CRAN

Date/Publication 2008-01-23 17:41:07

R topics documented:

arth800line	2
BuildEdges	3
BuildNetwork	5
DBNScoreStep1	8
DBNScoreStep2	10
PRcurve	13
ROCcurve	15
SimulGeneExpressionAR1	17
SimulNetworkAdjMatrix	18
Index	20

`arth800line`*Arabidopsis Thaliana temporal gene expression data*

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)

Lèbre Sophie (<http://www3.imperial.ac.uk/theoreticalgenomics/people/slebre>),
Chiquet Julien (<http://stat.genopole.cnrs.fr/~jchiquet/>).

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

Smith et al. 2004. Diurnal changes in the transcriptom encoding enzymes of starch metabolism provide evidence for both transcriptional and posttranscriptional regulation of starch metabolism in *Arabidopsis* leaves. *Plant Physiol.* 136: 2687-2699.

Examples

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

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

## plot first ten time series
plot(1:23,arth800line[,60],type="l",ylim=c(2,12), xlab="Time",
```

```

ylab="Log2 transformed expression",lwd=2,
main="Log2 transformed expression of a subset of genes of A. Thaliana")

color=1
for (i in idx){
  color=color+1
  lines(1:23,arth800line[,i,],col=color,lwd=2)
}

```

BuildEdges

*Edges listing and evaluation***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

```

out <- 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)

Lèbre Sophie (<http://www3.imperial.ac.uk/theoreticalgenomics/people/slebre/>),
 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 DBN

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

## building the edges of the Step1 inferred Graph
alpha1 <- 0.5
G1 <- BuildEdges(S1$S1ls,threshold=alpha1,dec=FALSE)
G1
## STEP 2- The second step score matrix
S2 <- DBNScoreStep2(S1$S1ls, Xn, method='ls', alpha1)

## building the edges of the Step2 inferred Graph
alpha2=0.05
G2 <- BuildEdges(S2,threshold=alpha2,dec=FALSE)
G2
```

```
## building the edges of the simulation Graph
Gsimul <- BuildEdges(MyNet$AdjMatrix,threshold=0,dec=TRUE)
Gsimul
```

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

```
out <- 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)

Lèbre Sophie (<http://www3.imperial.ac.uk/theoreticalgenomics/people/slebre>),
 Chiquet Julien (<http://stat.genopole.cnrs.fr/~jchiquet/>).

See Also

BuildEdges

Examples

```

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===== \n")
cat("SIMULATION\n\n")

## =====
## SIMULATING THE TIME SERIES EXPERIMENTS
## _____
##
## Autoregressive model
##
cat("Time series experiments with")

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

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

## STEP 2
## -----
cat("STEP 2...\n")

```

```

alpha1=0.5
S2 <- DBNScoreStep2(S1$S1ls, data=Xn, method='ls', alpha1=alpha1)

## =====
## POST TREATMENTS

## building the inferred Graph
G1 <- BuildEdges(S1$S1ls,threshold=alpha1,dec=FALSE)

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

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

## =====
## PLOTTING THE RESULTS...
## _____

cat("\n=====\\n")
cat("SUMMARY\\n\\n")
cat("Plotting the results...\\n")

## The Original graph and data
## -----
attach(MyNet)
pos <- gplot(abs(AdjMatrix), vertex.cex=1.5, diag=TRUE,
             displaylabel=TRUE, usecurv=TRUE,
             boxed.label=FALSE, main="Simulated network")
detach(MyNet)

## The Inferred Nets
## -----
#after Step 2

split.screen(c(1,2))
screen(1)
attach(Step2InferredNet)
gplot(abs(AdjMatrix), vertex.cex=1.5, diag=TRUE, coor=pos,
       displaylabel=TRUE, usecurv=TRUE,
       boxed.label=FALSE, main="Inferred network - Step 2")
detach(Step2InferredNet)

#after Step 1
screen(2)
attach(Step1InferredNet)
gplot(abs(AdjMatrix), vertex.cex=1.5, diag=TRUE, coor=pos,
       displaylabel=TRUE, usecurv=TRUE,
       boxed.label=FALSE, main="Inferred network - Step 1")
detach(Step1InferredNet)

```

```
close.screen(all = TRUE)

cat ("")
cat ("\nDONE !\n")
```

DBNScoreStep1 *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 $S1$ ($p \times p$) which contains the score of each edge of a Dynamic Bayesian Network (DAG $G(1)$) describing first order dependencies between successive 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

```
out <- 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', 'lsqr'), 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 $i = 1 \dots p$,

```
out_i <- DBNScoreStep1(data, target=i).
```

Author(s)

Lèbre Sophie (<http://www3.imperial.ac.uk/theoreticalgenomics/people/slebre/>),

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

References

Lèbre, S. 2007. Inferring Dynamic Bayesian Networks with low order dependencies. Preprint available at <http://hal.archives-ouvertes.fr/hal-00142109>.

See Also

DBNScoreStep2, BuildEdges, PRcurve.

Examples

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

data(arth800line)
data<-as.matrix(arth800line)
idx<-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=idx, predPosition=idx)
round(out$S1ls, 2)

alpha1=0.5
edgesG1idx<-BuildEdges(score=out$S1ls, threshold=alpha1,
                        targetNames=idx, predNames=idx, prec=6)
edgesG1names<-BuildEdges(score=out$S1ls, threshold=alpha1,
                          targetNames=names, predNames=names, prec=6)

edgesG1idx[1:15,]
edgesG1names[1:15,]

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

S2

alpha2=0.05
edgesGidx<-BuildEdges(score=S2, threshold=alpha2,
```

```

                                targetNames=idx,predNames=idx,prec=6)
edgesGidx

## As the number of genes is reduced to 10 here, this results slightly differ
## from the results obtained in the Preprint cited in References.

## encoding as the adjacency matrix graph
Step1InferredNet <- BuildNetwork(edgesGidx,idx)

## encoding as the adjancecy matrix graph
Step2InferredNet <- BuildNetwork(edgesGidx,idx)

## The Inferred Nets
## -----

#after Step 2
split.screen(c(1,2))
screen(1)
attach(Step2InferredNet)
pos<-gplot(t(AdjMatrix), vertex.cex=1.5, diag=TRUE,
           displaylabel=TRUE, usecurv=TRUE, label=names,
           boxed.label=FALSE, main="Inferred network - Step 2")
detach(Step2InferredNet)

#after Step 1
screen(2)
attach(Step1InferredNet)
gplot(t(AdjMatrix), vertex.cex=1.5, diag=TRUE, coord=pos,
      displaylabel=TRUE, usecurv=TRUE, label=names,
      boxed.label=FALSE, main="Inferred network - Step 1")
detach(Step1InferredNet)
close.screen(all = TRUE)

```

DBNScoreStep2

*Full order dependence DAG G score matrix inference from a 1st order
dependence score matrix S1*

Description

Given a time series dataset for p genes, a 1st order dependence score matrix $S1$ (obtained with function `DBNScoreStep1`) and a threshold α_1 for edge selection in matrix $S1$, 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

```
out<-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 rows, d columns) 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)

Lèbre Sophie (<http://www3.imperial.ac.uk/theoreticalgenomics/people/slebre>),
 Chiquet Julien (<http://stat.genopole.cnrs.fr/~jchiquet/>).

References

Lèbre, S. 2007. Inferring Dynamic Bayesian Networks with low order dependencies. Preprint available at <http://hal.archives-ouvertes.fr/hal-00142109>.

See Also

`DBNScoreStep1`, `BuildEdges`.

Examples

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

data(arth800line)
data<-as.matrix(arth800line)
```

```

idx<-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=idx,predPosition=idx)
round(out$S1ls,2)

alpha1=0.5
edgesG1idx<-BuildEdges(score=out$S1ls,threshold=alpha1,
                        targetNames=idx,predNames=idx,prec=6)
edgesG1names<-BuildEdges(score=out$S1ls,threshold=alpha1,
                          targetNames=names,predNames=names,prec=6)
edgesG1idx[1:15,]
edgesG1names[1:15,]

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

alpha2=0.05
edgesGidx<-BuildEdges(score=S2,threshold=alpha2,
                      targetNames=idx,predNames=idx,prec=6)
edgesGidx

## As the number of genes is reduced to 10 here, this results slightly differ
## from the results obtained in the Preprint cited in References.

## encoding as the adjacency matrix graph
Step1InferredNet <- BuildNetwork(edgesG1idx,idx)

## encoding as the adjancecy matrix graph
Step2InferredNet <- BuildNetwork(edgesGidx,idx)

## The Inferred Nets
## -----

split.screen(c(1,2))

#after Step 2
screen(1)
attach(Step2InferredNet)
pos<-gplot(t(AdjMatrix), vertex.cex=1.5, diag=TRUE,
           displaylabel=TRUE, usecurv=TRUE, label=names,
           boxed.label=FALSE, main="Inferred network - Step 2")
detach(Step2InferredNet)

#after Step 1
screen(2)
attach(Step1InferredNet)

```

```
gplot(t(AdjMatrix), vertex.cex=1.5, diag=TRUE, coord=pos,
      displaylabel=TRUE, usecurv=TRUE, label=names,
      boxed.label=FALSE, main="Inferred network - Step 1")
detach(Step1InferredNet)
close.screen(all = TRUE)
```

PRcurve

PR curves computation

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

```
out <- 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)

Lèbre Sophie (<http://www3.imperial.ac.uk/theoreticalgenomics/people/slebre>),
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(-1,0.5,0.5,1))

## 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 <- SimulGeneExpressionAR1(MyNet$A,B,X0,sigmaEps,n)

## compute score S1
out<-DBNScoreStep1(Xn)
pr1<-PRcurve(score=out$S1ls,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

## compute score S2 from S1
## depending on the generated data, the threshold alpha1 has to be chosen differently.
alpha1=0.8
S2<-DBNScoreStep2(S1=out$S1ls,data=Xn,alpha1=alpha1)
pr2_0.8<-PRcurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

alpha1=0.6
S2<-DBNScoreStep2(S1=out$S1ls,data=Xn,alpha1=alpha1)
pr2_0.6<-PRcurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

alpha1=0.4
S2<-DBNScoreStep2(S1=out$S1ls,data=Xn,alpha1=alpha1)
pr2_0.4<-PRcurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

alpha1=0.2
S2<-DBNScoreStep2(S1=out$S1ls,data=Xn,alpha1=alpha1)
pr2_0.2<-PRcurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

plot(pr1$recall,pr1$precision,type="l", main="PR curves after both Step1
and Step2",
      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 (alpha=0.8)", "Step 2 (alpha=0.6)",
      "Step 2 (alpha=0.4)", "Step 2 (alpha=0.2)")
legend(0,0.265, leg, lty=c(2,1,1,1,1), col=c(1,3,4,5,6),lwd=array(2,5))

```

ROCcurve

ROC curves computation

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

```
out<-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)

Lèbre Sophie (<http://www3.imperial.ac.uk/theoreticalgenomics/people/slebre>),
 Chiquet Julien (<http://stat.genopole.cnrs.fr/~jchiquet/>).

See Also

DBNScoreStep1, DBNScoreStep2, PRcurve.

Examples

```

## 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 Xt
X0 <- B + rnorm(p,0,sigmaEps*10)
## number of time points
n <- 20

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

## compute score S1
out<-DBNScoreStep1(Xn)
roc1<-ROCcurve(score=out$S1ls,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

## compute score S2 from S1
## depending on the generated data, the threshold alpha1 has to be chosen differently.
alpha1=0.8
S2<-DBNScoreStep2(S1=out$S1ls,data=Xn,alpha1=alpha1)
roc2_0.8<-ROCcurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

alpha1=0.6
S2<-DBNScoreStep2(S1=out$S1ls,data=Xn,alpha1=alpha1)
roc2_0.6<-ROCcurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

alpha1=0.4
S2<-DBNScoreStep2(S1=out$S1ls,data=Xn,alpha1=alpha1)
roc2_0.4<-ROCcurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

alpha1=0.2
S2<-DBNScoreStep2(S1=out$S1ls,data=Xn,alpha1=alpha1)
roc2_0.2<-ROCcurve(score=S2,validMat=abs(MyNet$AdjMatrix)>0,dec=FALSE)

TP=sum(abs(MyNet$AdjMatrix)>0)
FN=p^2-TP

plot(roc1$x/FN,roc1$y/TP,type="l", main="ROC curve after both Step1 and
Step2",
      ylab="True Positive Rate", xlab="False Negative Rate",lwd=2,lty=2)
lines(roc2_0.8$x/FN, roc2_0.8$y/TP, col=3,lwd=2)
lines(roc2_0.6$x/FN, roc2_0.6$y/TP, col=4,lwd=2)
lines(roc2_0.4$x/FN, roc2_0.4$y/TP, col=5,lwd=2)
lines(roc2_0.2$x/FN, roc2_0.2$y/TP, 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 (alpha=0.8)", "Step 2 (alpha=0.6)",
      "Step 2 (alpha=0.4)", "Step 2 (alpha=0.2)")
legend(0.568,0.265, leg, lty=c(2,1,1,1,1), col=c(1,3,4,5,6),lwd=array(2,5))

```

SimulGeneExpressionAR1

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

```
out<-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)

Lèbre Sophie (<http://www3.imperial.ac.uk/theoreticalgenomics/people/slebre>),
 Chiquet Julien (<http://stat.genopole.cnrs.fr/~jchiquet/>).

See Also

SimulNetworkAdjMatrix

Examples

```

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 Xt
X0 <- B + rnorm(p,0,sigmaEps*10)
## number of time points
n <- 20

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

```

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

```
out<-SimulNetworkAdjMatrix(Num,EdgesProp,Range,Labels=1:Num)
```

Arguments

Num	number of genes
EdgesProp	edges proportion in the network
Range	vector with 4 elements specifying range values for the adjacency matrix generation (minimum negative value, maximum negative value, minimum positive value, maximum positive value)
Labels	an optional vector of labels for the edges

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)

Lèbre Sophie (<http://www3.imperial.ac.uk/theoreticalgenomics/people/slebre/>),
Chiquet Julien (<http://stat.genopole.cnrs.fr/~jchiquet>).

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, [4](#)
 - SimulGeneExpressionAR1, [16](#)
 - SimulNetworkAdjMatrix, [17](#)
 - *Topic **datasets**
 - arth800line, [1](#)
 - *Topic **graphs**
 - DBNScoreStep1, [7](#)
 - DBNScoreStep2, [10](#)
 - *Topic **models**
 - DBNScoreStep1, [7](#)
 - DBNScoreStep2, [10](#)
 - *Topic **regression**
 - DBNScoreStep1, [7](#)
 - DBNScoreStep2, [10](#)
 - *Topic **robust**
 - DBNScoreStep1, [7](#)
 - DBNScoreStep2, [10](#)
 - *Topic **ts**
 - DBNScoreStep1, [7](#)
 - DBNScoreStep2, [10](#)
 - SimulGeneExpressionAR1, [16](#)
 - *Topic **utilities**
 - BuildEdges, [2](#)
 - PRcurve, [12](#)
 - ROCcurve, [14](#)
- arth800line, [1](#)
- BuildEdges, [2](#)
- BuildNetwork, [4](#)
- DBNScoreStep1, [7](#)
- DBNScoreStep2, [10](#)
- PRcurve, [12](#)
- ROCcurve, [14](#)
- SimulGeneExpressionAR1, [16](#)
- SimulNetworkAdjMatrix, [17](#)