Package 'interventionalDBN'

February 20, 2015

Type Package

Title Interventional Inference for Dynamic Bayesian Networks

Version 1.2.2

Date 2014-01-08

Author Simon Spencer

Maintainer Simon Spencer <s.e.f.spencer@warwick.ac.uk>

Description This package allows a dynamic Bayesian network to be inferred from microarray time-course data with interventions (inhibitors).

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-05-30 16:19:24

R topics documented:

interventionalDBN-package

Interventional Inference for Dynamic Bayesian Networks

Description

This package allows a dynamic Bayesian network to be inferred from microarray timecourse data with interventions (inhibitors).

Details

Package:	interventionalDBN
Type:	Package
Version:	1.2.2
Date:	2014-01-03
License:	GPL version 2 or newer
LazyLoad:	yes

Includes functions for formating the data (formatData), estimating the effects of an intervention (interventionEffects) and performing network inference (interventionalInference).

Author(s)

Simon Spencer

Maintainer: Simon Spencer <s.e.f.spencer@warwick.ac.uk>

See Also

interventionalInferenceAdvanced, countGraphs, interventionalData, linesROC, nxt, trueMatrix, warshall.

Examples

```
library(interventionalDBN)
data(interventionalData)# loads interventionalData.
# Load your own data spreadsheet using myData<-read.csv("myDataFile.csv").

# Estimate nodes downstream of intervention.
egfriEffects<-interventionEffects(interventionalData,1,"DMSO","EGFRi")
aktiEffects <-interventionEffects(interventionalData,1,"DMSO","AKTi")

# Format the data for network inference
d<-formatData(interventionalData)

# EGFRi is active in conditions 2 and 4, AKTi is active in conditions 3 and 4.
# Each condition has 8 timepoints.
```

```
Z<-matrix(0,32,15)
Z[9:16,1]<-1 # EGFR (node 1) is inhibited in condition 2
Z[25:32,1]<-1 # EGFR (node 1) is inhibited in condition 4
Z[17:24,8]<-1 # AKT (node 8) is inhibited in condition 3
Z[25:32,8]<-1 # AKT (node 8) is inhibited in condition 4

# Perform network inference
myNetwork<-interventionalInference(d$y,d$X0,d$X1,Z,max.indeg=3,
  perfectOut=TRUE,fixedEffectOut=TRUE)

# Make ROC curve, to see how well we have done.
data(trueMatrix)
plot(0:1,0:1,t="l",col="grey",xlab="False positive rate",ylab="False negative rate",
  main="ROC curve showing network inference performance.")
redArea<-linesROC(trueMatrix,myNetwork$pep) # ROC area is also sent to the console.

# More realistically, the true edge matrix is unknown.
# We can use descendancy to get (a much coarser) ROC,
# which is based only on nodes that are downstream of the inhibitors.
pap<-warshall(myNetwork$pep)
effectMatrix<-matrix(NA,15,15)
effectMatrix[1,]<-1*(egfriEffects$p.values<=0.1)
effectMatrix[8,]<-1*( aktiEffects$p.values<=0.1)
blueArea<-linesROC(effectMatrix,myNetwork$pep,col="blue")
legend("bottomright",c("Edge matrix known","Descendancy ROC"),col=c("red","blue"),lty=1)
```

`countGraphs` *Count the number of possible parents*

Description

Counts the number of choices of parents given a maximum in-degree restriction.

Usage

```
countGraphs(nodes, max.indeg)
```

Arguments

`nodes`	A positive integer specifying the number of nodes in the network.
`max.indeg`	A positive integer specifying the in-degree restriction.

Details

Nodes can be a parent to themselves. The number of possible networks is given by:
nodes*countGraphs(nodes,max.indeg)

Value

Returns an integer given by $\sum_{i=0}^{m} \binom{n}{i}$, where `nodes`$= n$ and `max.indegree`$= m$

Author(s)

Simon Spencer

See Also

interventionalInference, interventionalDBN-package

Examples

```
countGraphs(10,3) # 176, the number of possible parent sets for each node.
10*countGraphs(10,3) # 1760, the total number of possible networks.
```

---

formatData *Format a microarray spreadsheet ready for interventional network inference function*

Description

This function formats a microarray timecourse dataset ready for the interventionalInference function.

Usage

```
formatData(d, cellLines = NULL, inhibitors = NULL, stimuli = NULL, times = NULL,
  nodes = NULL, intercept = TRUE, initialIntercept = TRUE, gradients = FALSE)
```

Arguments

d	A microarray spreadsheet, a $samples$ by (4 + P) matrix, where P is the number of measurements for each sample. Column 1 gives the cell line in each sample. Column 2 gives the inhibitor used in each sample. Column 3 gives the stimulus used in each sample. Column 4 gives the time each sample was measured.
cellLines	A vector specifying a subset of cell lines to analyse (if absent, they are all used).
inhibitors	A vector specifying a subset of the inhibitors to analyse (if absent, they are all used).
stimuli	A vector specifying a subset of the stimuli to analyse (if absent, they are all used).
times	A vector specifying a subset of the times to analyse as the response (if absent, they are all used).
nodes	A vector specifying the indices of a subset of nodes to include in the analysis. Further nodes can be removed from the response in the interventionalInferenceDBN function.
intercept	A logical value indicating whether an intercept parameter should be included in all models.

`initialIntercept`
: A logical value indicating whether an intercept parameter should be used to estimate the level at the initial timepoint. Only used if the initial timepoint is in the response.

`gradients`
: A logical value indicating whether the concentraion gradient should be used as the response instead of the raw concentrations. This model has parallels with a dynamical systems viewpoint, and requires the covariance matrix to be adjusted. See `Sigma`.

Details

The entries of column 4 of `d` must be real numbers. Missing values are acceptable and are handled as follows:

1. Missing values in the response are ignored.
2. For the predictors, if a single timepoint is missing, the predictors are interpolated from the two immediate neighbours.
3. If one of the two immediate neighbours is missing then the response is ignored.
4. UNLESS the predictor in question is for the initial observation (which is always missing), in which case 0 is returned, so that the level at zero can be estimated by a second intercept parameter in the `interventionalInferenceDBN` function.

Value

`y`
: The n by P response matrix, where n is the number of observations in the response. Not necesarily the same as the number of samples.

`X0`
: The n by a design matrix of predcitors to be included in all models. Usually the intercept and zero intercept (if present).

`X1`
: The n by P design matrix of predictors to undergo model selection.

`Sigma`
: The n by n covariance matrix for a single column of `y` (proportional to σ^2). The identity matrix, unless `gradients` is `TRUE`.

`sampleInfo`
: An n by 4 matrix giving the cell line, inhibitor, stimulus and timepoint for each observation used in the response.

`interpolated`
: A matrix similar to `sampleInfo`, giving the particulars of any observations for which the predictors were interpolated. Empty if no interpolation has been used.

`cond`
: A vector indexing the experimental conditions, given by the cell line, inhibitor and stimulus used in each sample.

Author(s)

Simon Spencer

See Also

`interventionalInference`, `interventionalInferenceAdvanced`, `interventionalDBN-package`, `interventionEffects`

Examples

```
data(interventionalData)
# Load your own data spreadsheet using myData<-read.csv("myDataFile.csv").

# Use everything
fullData <- formatData(interventionalData)

# Use only DMSO and EGFRi samples.
halfData <- formatData(interventionalData,inhibitors=c("DMSO","EGFRi"))

# Produce gradients as response
diffData <- formatData(interventionalData,gradients=TRUE,initialIntercept=FALSE)
# Different results if we use the time between observations, rather than the timepoint.
interventionalData[,4]<-rep(c(0,5,10,20,30,60,90,120),4)
diffData2 <- formatData(interventionalData,gradients=TRUE,initialIntercept=FALSE)

# When there is missing data, interpolation also uses the time differences.
missingData <- interventionalData[-4,]
fullData2 <- formatData(missingData)
```

---

interventionalData *Simulated micro-array timecourse data speadsheet.*

Description

A simulated microarray timecourse dataset, generated using the perfect and fixed effect intervention models.

Usage

```
data(interventionalData)
```

Format

A data frame with 32 observations on the following 19 variables.

`Cell.line` a factor with levels representing the cell line.

`Inhibitor` a factor with levels describing the inhibitors used in each sample.

`Stimuli` a factor with levels describing the stimulus used in each sample.

`Timepoint` a integer vector (starting from zero) representing the time index of each sample.

`EGFR` The remaining columns give the log-concentrations of each node.

`SRC`

`STAT5`

`Mek`

`MAPK`

`p90RSK`

PDK

AKT

GSK

TSC2

BAD

mTOR

p70S6K

S6

FOXO3

Source

Simulated by Simon Spencer.

See Also

formatData, interventionEffects, interventionalDBN-package.

Examples

```
data(interventionalData)
interventionalData
```

---

interventionalInference

Dynamic Bayesian Network inference with interventions.

Description

This function performs exact Bayesian inference for dynamic Bayesian networks using microarray timecourse data. Several intervention models can be chosen to take into account the effect of inhibitors.

Usage

```
interventionalInference(y, X0, X1, Z, max.indeg,
  g = NULL, Sigma = NULL, inferParents = NULL, allowSelfEdges = TRUE,
  perfectOut = FALSE, fixedEffectOut = FALSE, mechanismChangeOut = FALSE,
  perfectIn = FALSE, fixedEffectIn = FALSE, mechanismChangeIn = FALSE,
  priorType = "uninformed", priorGraph = NULL, priorStrength = 3,
  fittedValues = FALSE)
```

Arguments

y	an n by P matrix filled with the response values, where n is the number of observations and P is the number of nodes.
X0	an n by a matrix - the part of the design matrix that is the same for all models. a is the number of parameters that are in all of the modesl.
X1	an n by P matrix - the part of the design matrix to undergo model selection. `colnames(X1)` provides the labels for the output.
Z	an n by P binary matrix. Entry i, j is one if node j is inhibited in sample i.
max.indeg	The maximum permitted in-degree for each node.
g	The constant g in Zellner's g-prior. Defaults to n.
Sigma	an n by n covariance matrix of the responses, divided by σ^2. Faster if not specified, in which case the identity matrix is assumed.
inferParents	a vector of node indices specifying which nodes to infer parents for. If omitted, parents are inferred for all nodes.
allowSelfEdges	Should self-edges be allowed?
perfectOut	Apply perfect-out interventions?
fixedEffectOut	Apply fixed-effect-out interventions?
mechanismChangeOut	Apply mechanism-change-out interventions? Note: cannot be applied with perfect interventions.
perfectIn	Apply perfect-in interventions?
fixedEffectIn	Apply fixed-effect-in interventions?
mechanismChangeIn	Apply mechanism-change-in interventions? Note: cannot be applied with perfect interventions.
priorType	One of `"uninformed"`, `"Mukherjee"` and `"Hamming"`. In the structural Hamming distance prior, each difference from the edges in `priorGraph` incurs a prior penalty of `exp(-priorStrength)`. In the Mukherjee-Speed prior, adding edges from outside `priorGraph` earns the same penalty as before, but if a prior edge is omitted a penalty is no longer incurred.
priorGraph	A P by P binary matrix specifying the prior graph. If $(i, j) = 1$ then node i influences node j. If omitted, an uninformed prior is used.
priorStrength	The prior strength parameter. Ignored (but don't set it to NA) if `priorGraph` is `NULL`. If specified as a vector then the value from that gives the highest marginal likelihood is chosen (Empirical Bayes).
fittedValues	Perform a second pass to calculate the fitted values?

Details

This function performs interventional inference with both -in and -out forms of the interventions. The targets of the interventions are specified in the matrix `Z`. This assumes that each node is the target of only one intervention - if this is not the case, you must use the `interventionalInferenceAdvanced` function. Certain combinations of interventions do not work together, in particular mixtures of

perfect and mechanism change interventions. Perfect-in and perfect-out can be used together. Mechanism-change-in and mechanism-change-out could potentially be used together, but are not currently implemented.

Value

`pep` A P by P matrix of posterior probabilities, where element (i, j) gives the posterior probability that node i influences node j.

`MAP` A P by P binary matrix giving the maximum a posteriori network.

`parentSets` A `countGraphs(P,max.indeg)` by P binary matrix, where element $(m, p$=1) iff node i is a parent in model m.

`ll` A `countGraphs(P,max.indeg)` by P matrix, where element (m, p) gives the log-likelihood for model m for node p.

`lpost` A `countGraphs(P,max.indeg)` by P matrix, where element (m, p) gives the log-posterior probability for model m for node p.

`MAPprob` A P vector where element p gives the posterior probability of the maximum a posteriori model for node p.

`MAPmodel` A P vector where element p gives the index of the maximum a posterior model for node p (between 1 and `countGraphs(P,max.indeg)`.

`marginal.likelihood`
A P by `length(priorStrength)` matrix that gives the marginal likelihood for each node.

`ebPriorStrength`
Value of `priorStrength` with the largest marginal likelihood, if `priorStrength` is a vector; `NULL` otherwise.

`yhat` The posterior expected fitted values, if `fittedValues` is TRUE.

`inputs` A list containing the inputs to `interventionalInference`

Author(s)

Simon Spencer

References

Spencer, S.E.F, Hill, S.M. and Mukherjee, S. (2012) Dynamic Bayesian networks for interventional data. CRiSM pre-print 12-24.
Mukherjee, S. and Speed, T.P. Network inference using informative priors. Proc. Nat. Acad. Sci. USA, 105, 14313-14318.

See Also

`interventionalDBN-package`, `formatData`

Examples

```
library(interventionalDBN)
data(interventionalData)# loads interventionalData.
# Load your own data spreadsheet using myData<-read.csv("myDataFile.csv").

# Format the data for network inference
d<-formatData(interventionalData)

# Perform network inference without modelling interventions.
myNetwork0<-interventionalInference(d$y,d$X0,d$X1,max.indeg=3,fittedValues=TRUE)

# EGFRi is active in conditions 2 and 4, AKTi is active in conditions 3 and 4.
# Each condition has 8 timepoints.
Z<-matrix(0,32,15)
Z[9:16,1]<-1 # EGFR (node 1) is inhibited in condition 2
Z[25:32,1]<-1 # EGFR (node 1) is inhibited in condition 4
Z[17:24,8]<-1 # AKT (node 8) is inhibited in condition 3
Z[25:32,8]<-1 # AKT (node 8) is inhibited in condition 4

# Perform network inference with perfect-out and fixed-effect-out interventions.
myNetwork1<-interventionalInference(d$y,d$X0,d$X1,Z,max.indeg=3,
  perfectOut=TRUE,fixedEffectOut=TRUE)

# Perform network inference on with mechanism-change-out interventions.
myNetwork2<-interventionalInference(d$y,d$X0,d$X1,Z,max.indeg=3,
  mechanismChangeOut=TRUE)

# Perform network inference with Mukherjee Prior that prefers to omit self-edges.
myNetwork3<-interventionalInference(d$y,d$X0,d$X1,Z,max.indeg=3,
  perfectOut=TRUE,fixedEffectOut=TRUE,
  priorType="Mukherjee",priorGraph=matrix(1,15,15)-diag(rep(1,15)),priorStrength=2)
# Compare with self-edge peps with myNetwork1
diag(myNetwork1$pep)-diag(myNetwork3$pep)

# Perform network inference with Hamming Prior that prefers self-edges,
# and use Empirical Bayes to choose the priorStrength.
myNetwork4<-interventionalInference(d$y,d$X0,d$X1,Z,max.indeg=3,
  perfectOut=TRUE,fixedEffectOut=TRUE,
  priorType="Hamming",priorGraph=diag(rep(1,15)),priorStrength=0:10/2)
# You should always check to see if the Empirical Bayes appears to be working.
plotMaxML(myNetwork4)

# Now let's try using using the gradients as the response.
# Note that we have to tranfser Sigma this time, as it is no longer the identity.
d<-formatData(interventionalData,gradients=TRUE,initialIntercept=FALSE)
# There are now only 28 observations
Z<-Z[c(2:8,10:16,18:24,26:32),]

# Perform network inference on gradients with perfect-in interventions.
myNetwork5<-interventionalInference(d$y,d$X0,d$X1,Z,max.indeg=3,
  Sigma=d$Sigma,perfectIn=TRUE,fittedValues=TRUE)
```

```
# Perform network inference on gradients with perfect-in and -out plus fixed-effect out.
myNetwork6<-interventionalInference(d$y,d$X0,d$X1,Z,max.indeg=3,
  Sigma=d$Sigma,perfectIn=TRUE,perfectOut=TRUE)
```

`interventionalInferenceAdvanced` *Dynamic Bayesian Network inference with interventions.*

Description

This function performs exact Bayesian inference for dynamic Bayesian networks using microarray timecourse data. Several intervention models can be chosen to take into account the effect of inhibitors.

Usage

```
interventionalInferenceAdvanced(y, X0, X1, cond, inhibition, inhibitors, max.indeg,
  g = NULL, Sigma = NULL, inferParents = NULL, allowSelfEdges = TRUE,
  perfect = FALSE, fixedEffect = FALSE, mechanismChange = FALSE,
  priorType = "uninformed", priorGraph = NULL, priorStrength = 3,
  fittedValues = FALSE)
```

Arguments

`y`	an n by P matrix filled with the response values, where n is the number of observations and P is the number of nodes.
`X0`	an n by a matrix - the part of the design matrix that is the same for all models. a is the number of parameters that are in all of the modesl.
`X1`	an n by P matrix - the part of the design matrix to undergo model selection. `colnames(X1)` provides the labels for the output.
`cond`	an n by 1 matrix giving the experimental condition number of each sample. Filled with integers from 1 to the number of different conditions.
`inhibition`	a $conditions$ by $inhibitors$ binary matrix, where element (c, i) is one iff inhibitor i is active in condition c.
`inhibitors`	an $inhibitors$ by P binary matrix, where element (i, p) is one iff inhibitor i affects node p.
`max.indeg`	The maximum permitted in-degree for each node.
`g`	The constant g in Zellner's g-prior. Defaults to n.
`Sigma`	an n by n covariance matrix of the responses, divided by σ^2. Faster if not specified, in which case the identity matrix is assumed.
`inferParents`	a vector of node indices specifying which nodes to infer parents for. If omitted, parents are inferred for all nodes.
`allowSelfEdges`	Should self-edges be allowed?
`perfect`	Apply perfect-out interventions?

`fixedEffect` Apply fixed-effect-out interventions?

`mechanismChange`
Apply mechanism-change-out interventions? Note: cannot be applied with perfect interventions.

`priorType` One of `"uninformed"`, `"Mukherjee"` and `"Hamming"`. In the structural Hamming distance prior, each difference from the edges in `priorGraph` incurs a prior penalty of `exp(-priorStrength)`. In the Mukherjee-Speed prior, adding edges from outside `priorGraph` earns the same penalty as before, but if a prior edge is omitted a penalty is no longer incurred.

`priorGraph` A P by P binary matrix specifying the prior graph. If $(i, j) = 1$ then node i influences node j. If omitted, an uninformed prior is used.

`priorStrength` The prior strength parameter. Ignored (but don't set it to NA) if `priorGraph` is `NULL`. If specified as a vector then the value from that gives the highest marginal likelihood is chosen (Empirical Bayes).

`fittedValues` Perform a second pass to calculate the fitted values?

Details

The function `interventionalInference` provides a simpler, but less general way of coding which inhibitors are active in each condition. Currently this advanced version only supports -out forms of the interventions. By default the fixed effects in the fixedEffect intervention are assumed to be additive in samples with multiple inhibitors. However if you do not wish for this to be the case, then you can simply define a dummy inhibitor for each combination of inhibitors and a new fixed effect parameter will be estimated. See example 7 below.

Value

`pep` A P by P matrix of posterior probabilities, where element (i, j) gives the posterior probability that node i influences node j.

`MAP` A P by P binary matrix giving the maximum a posteriori network.

`parentSets` A `countGraphs(P,max.indeg)` by P binary matrix, where element $(m, p$=1) iff node i is a parent in model m.

`ll` A `countGraphs(P,max.indeg)` by P matrix, where element (m, p) gives the log-likelihood for model m for node p.

`lpost` A `countGraphs(P,max.indeg)` by P matrix, where element (m, p) gives the log-posterior probability for model m for node p.

`MAPprob` A P vector where element p gives the posterior probability of the maximum a posteriori model for node p.

`MAPmodel` A P vector where element p gives the index of the maximum a posterior model for node p (between 1 and `countGraphs(P,max.indeg)`.

`marginal.likelihood`
A P by `length(priorStrength)` matrix that gives the marginal likelihood for each node.

`ebPriorStrength`
Value of `priorStrength` with the largest marginal likelihood, if `priorStrength` is a vector; `NULL` otherwise.

yhat	The posterior expected fitted values, if fittedValues is TRUE.
inputs	A list containing the inputs to interventionalInferenceAdvanced

Author(s)

Simon Spencer

References

Spencer, S.E.F, Hill, S.M. and Mukherjee, S. (2012) Dynamic Bayesian networks for interventional data. CRiSM pre-print 12-24.
Mukherjee, S. and Speed, T.P. Network inference using informative priors. Proc. Nat. Acad. Sci. USA, 105, 14313-14318.

See Also

interventionalDBN-package, interventionalInference, formatData

Examples

```
library(interventionalDBN)
data(interventionalData)# loads interventionalData.
# Load your own data spreadsheet using myData<-read.csv("myDataFile.csv").

# Format the data for network inference
d<-formatData(interventionalData)

# Perform network inference without modelling interventions.
myNetwork0<-interventionalInferenceAdvanced(d$y,d$X0,d$X1,max.indeg=3,fittedValues=TRUE)

# EGFRi is active in conditions 2 and 4, AKTi is active in conditions 3 and 4.
myInhibition<-cbind(c(0,1,0,1),c(0,0,1,1))
myInhibitors<-matrix(0,2,15)
myInhibitors[1,1]<-1 # EGFRi targets EGFR (node 1).
myInhibitors[2,8]<-1 # AKTi targets AKT (node 8).

# Perform network inference with perfect and fixed effect interventions.
myNetwork1<-interventionalInferenceAdvanced(d$y,d$X0,d$X1,d$cond,max.indeg=3,
  inhibition=myInhibition,inhibitors=myInhibitors,perfect=TRUE,fixedEffect=TRUE)

# Perform network inference on with mechanism change interventions.
myNetwork2<-interventionalInferenceAdvanced(d$y,d$X0,d$X1,d$cond,max.indeg=3,
  inhibition=myInhibition,inhibitors=myInhibitors,mechanismChange=TRUE)

# Perform network inference with Mukherjee Prior that prefers to omit self-edges.
myNetwork3<-interventionalInferenceAdvanced(d$y,d$X0,d$X1,d$cond,max.indeg=3,
  inhibition=myInhibition,inhibitors=myInhibitors,perfect=TRUE,fixedEffect=TRUE,
  priorType="Mukherjee",priorGraph=matrix(1,15,15)-diag(rep(1,15)),priorStrength=2)
# Compare with self-edge peps with myNetwork1
diag(myNetwork1$pep)-diag(myNetwork3$pep)

# Perform network inference with Hamming Prior that prefers self-edges,
```

```
# and use Empirical Bayes to choose the priorStrength.
myNetwork4<-interventionalInferenceAdvanced(d$y,d$X0,d$X1,d$cond,max.indeg=3,
  inhibition=myInhibition,inhibitors=myInhibitors,perfect=TRUE,fixedEffect=TRUE,
  priorType="Hamming",priorGraph=diag(rep(1,15)),priorStrength=0:10/2)
# You should always check to see if the Empirical Bayes appears to be working.
plotMaxML(myNetwork4)

# Now let's try using using the gradients as the response.
# Note that we have to tranfser Sigma this time, as it is no longer the identity.
d<-formatData(interventionalData,gradients=TRUE,initialIntercept=FALSE)

# Perform network inference on gradients with perfect-out interventions.
myNetwork5<-interventionalInferenceAdvanced(d$y,d$X0,d$X1,d$cond,max.indeg=3,
  Sigma=d$Sigma,inhibition=myInhibition,inhibitors=myInhibitors,perfect=TRUE)

# So far we have assumed that the fixed effects are additive in EGFRi+AKTi.
# Now let's change this, by coding EGFRi+AKTi as a separate inhibitor.
d<-formatData(interventionalData)
# EGFRi+AKTi is active in condition 4.
myInhibition<-cbind(c(0,1,0,0),c(0,0,1,0),c(0,0,0,1))
myInhibitors<-matrix(0,3,15)
myInhibitors[1,1]<-1 # EGFRi targets EGFR (node 1).
myInhibitors[2,8]<-1 # AKTi targets AKT (node 8).
myInhibitors[3,c(1,8)]<-1 # EGFRi+AKTi targets both.

# Perform network inference on gradients with fixed effect interventions.
myNetwork7<-interventionalInferenceAdvanced(d$y,d$X0,d$X1,d$cond,max.indeg=3,
  inhibition=myInhibition,inhibitors=myInhibitors,fixedEffect=TRUE)
```

`interventionEffects` *Calculate interventional effects*

Description

This function assesses which nodes are downstream of the nodes that are the target of the interventions. The samples are assumed to be independent, and the difference between the inhibited and baseline concentrations is assumed to be Gaussian. This leads to a t-distribution for the mean difference across the timecourse.

Usage

```
interventionEffects(d, cellLine, baseline, inhibited)
```

Arguments

d	A microarray spreadsheet, a $samples$ by (4 + P) matrix, where P is the number of measurements for each sample. Column 1 gives the cell line in each sample. Column 2 gives the inhibitor used in each sample.

Column 3 gives the stimulus used in each sample.
Column 4 gives the time each sample was measured.

`cellLine` The cell line to investigate (must match an entry in column 1 of `d`). Must be specified even if there is only one.

`baseline` The baseline inhibition condition (must match an entry in column 2 of `d`).

`inhibited` The active inhibition condition (must match an entry in column 2 of `d`).

Details

The function performs a t-test for each stimuli seperately as well as for all the stimuli combined together, which may be less reliable because the assumptions are stronger.

Value

`n.differences` A vector giving the number of differences used to calculate the t-statistic for each stimulus.

`t.statistics` A vector of t-statistics for the stimuli separately.

`degrees.freedom`
The corresponding vector of degrees of freedom for each test.

`p.values` The corresponding vector of p-values.

`heatmap.p.values`
The corresponding vector of $sign(T)(1-p)$. This can make a nice heatmap, as significant increases and significant decreases in concentration are at opposite ends of the scale.

`all.stim.t.statistic`
The t-statistic for the stimuli combined.

`all.stim.degrees.freedom`
The degrees of freedom for the stimuli combined.

`all.stim.p.values`
The p-value for the stimuli combined.

`all.stim.heatmap.p.values`
$sign(T)(1-p)$ for all stimuli combined.

Author(s)

Simon Spencer

See Also

`formatData`, `interventionalDBN-package`

Examples

```
data(interventionalData)
effect1<-interventionEffects(interventionalData,1,"DMSO","EGFRi")
effect2<-interventionEffects(interventionalData,1,"DMSO","AKTi")
heats<-rbind(effect1$heatmap.p.values,effect2$heatmap.p.values)
```

```
image(heats, breaks=c(-1,-0.95,-0.9,0.9,0.95,1),
  col=c("red","darkred","black","darkgreen","green"),xaxt="n",yaxt="n",
  xlab="Green = up when inhibitor is present\nRed = down when inhibitor is present")
# Or use the package gplots for more colour graduation
#library("gplots")
#image(heats,breaks=c(-1,-0.999,-0.99,-0.975,-0.95,-0.9,0.9,0.95,0.975,0.99,0.999,1)
#  ,col=redgreen(11),xaxt="n",yaxt="n")
axis(1,0:1,c("EGFRi","AKTi"))
axis(2,0:14/14,colnames(effect1$p.values),las=1)
```

`linesROC` *Add an ROC curve to an existing plot.*

Description

A simple function to produce an ROC curve from a known edge matrix and a posterior edge probability matrix.

Usage

```
linesROC(trueMatrix, pep, col = "red", lty = 1, lwd = 1)
```

Arguments

`trueMatrix`	The 'true' edge matrix.
`pep`	A matrix of posterior edge probabilities.
`col`	A colour (passed to segments).
`lty`	A line type (passed to segments).
`lwd`	A line width (passed to segments).

Details

The area of the ROC curve is also sent to the console.

Value

The area of the ROC curve.

Author(s)

Simon Spencer

See Also

`interventionalDBN-package`

Examples

```
trueMatrix<-matrix(rbinom(225,1,0.5),15,15)
pep<-matrix(runif(225,0.2,1)*trueMatrix+runif(225,0,0.5)*(1-trueMatrix),15,15)
plot(0:1, 0:1, t="l", col="grey", xlab="False positive rate",
  ylab="False negative rate", main="An ROC curve.")
linesROC(trueMatrix,pep)
```

`nxt` *Produces the next set of parents from an existing set of parents (internal).*

Description

A function to find the next parent set in the sequence.

Usage

```
nxt(g, max.indeg)
```

Arguments

`g`	A binary vector of length `nodes`
`max.indeg`	The maximum in-degree of the network

Value

A different binary vector of length `nodes`

Author(s)

Simon Spencer

See Also

`countGraphs`,`interventionalDBN-package`

Examples

```
g<-rep(0,7)
for (i in 1:countGraphs(7,3)) {
  cat(g,"\n")
  g<-nxt(g,3)
}
```

plotMaxML *Plot the performance of maximum marginal likelihood (Empirical Bayes).*

Description

Make a plot of the marginal likelihood against the prior strength parameter, highlighting the value used to produce the network.

Usage

```
plotMaxML(output,xlab="Prior strength",ylab="Marginal likelihood",
  col.max="red",lty.max=3,lwd.max=1,...)
```

Arguments

output	The object returned from the interventionalInference function.
xlab	A label for the prior strength axis.
ylab	A label for the marginal likelihood axis.
col.max	The colour of the line highlighting the maximum.
lty.max	The line type of the highlight.
lwd.max	The line width of the highlight.
...	Other arguments, such as main, which are passed to plot.

Details

It is important to check that the Empirical Bayes calculation is doing something sensible.

Author(s)

Simon Spencer

See Also

interventionalDBN-package,interventionalInference

Examples

```
library(interventionalDBN)
data(interventionalData)# loads interventionalData.
# Load your own data spreadsheet using myData<-read.csv("myDataFile.csv").

# Format the data for network inference
d<-formatData(interventionalData)

# EGFRi is active in conditions 2 and 4, AKTi is active in conditions 3 and 4.
# Each condition has 8 timepoints.
```

```
Z<-matrix(0,32,15)
Z[9:16,1]<-1 # EGFR (node 1) inhibited in condition 2
Z[25:32,1]<-1 # EGFR inhibited in condition 4
Z[17:24,8]<-1 # AKT (node 8) inhibited in condition 3
Z[25:32,8]<-1 # AKT inhibited in condition 4

# Perform network inference with Hamming Prior that prefers self-edges,
# and use Empirical Bayes to choose the priorStrength.
myNetwork4<-interventionalInference(d$y,d$X0,d$X1,Z,max.indeg=3,
  perfectOut=TRUE,fixedEffectOut=TRUE,
  priorType="Hamming",priorGraph=diag(rep(1,15)),priorStrength=0:10/2)
# You should always check to see if the Empirical Bayes appears to be working.
plotMaxML(myNetwork4)
```

`trueMatrix` *The true edge matrix used to generate* `interventionalData`.

Description

The 15 by 15 binary edge matrix that was used to generate the dataset `interventionalData`.

Usage

```
data(trueMatrix)
```

Source

Simon Spencer

See Also

`interventionalData`,`interventionalDBN-package`

Examples

```
data(trueMatrix)
pep<-matrix(runif(225,0.2,1)*trueMatrix+runif(225,0,0.5)*(1-trueMatrix),15,15)
plot(0:1, 0:1, t="l", col="grey", xlab="False positive rate",
  ylab="False negative rate",main="An ROC curve.")
linesROC(trueMatrix,pep)
```

`warshall` *Find the largest edge probability threshold that connects a pair of nodes.*

Description

This function runs a slight variation on the Warshall algorithm to find the largest posterior edge probability threshold that allows each pair of nodes to remain connected. It is useful for calculating ROC curves based on descendancy information.

Usage

```
warshall(M)
```

Arguments

`M`	A square matrix of probabilities.

Details

The Warshall algorithm is $O(P)^3$, where P is the number of nodes.

Value

A square matrix, where element (i, j) is the largest edge probability threshold that allows i to remain connected to j.

Author(s)

Simon Spencer

See Also

`interventionEffects`, `interventionalDBN-package`

Examples

```
M1<-rbind(c(0.5,1,0),c(0,0,1),c(0,0,0))# A->B->C
warshall(M1)# A is upstream of B and C, B is upstream of C.
# Note that A is upstream of itself iff there is a cycle.

M2<-matrix(runif(25),5,5)
warshall(M2)
```

Index