R documentation

of 'Binference.Rd' etc.

April 11, 2012

`Binference` *Bayesian network inference*

Description

This function uses data (CNOlist) to infer a Bayesian network using the catnet package.

Usage

```
Binference(CNOlist, Model="AIC", tempCheckOrders=10,
            maxIter=100, filename="BAYESIAN")
```

Arguments

`CNOlist`	a CNOlist structure, as produced by makeCNOlist
`mode`	a character, optimization network selection criterion such as "AIC" and "BIC", to be used in cnSearchSA
`tempCheckOrders`	an integer, the number of iteration, orders to be searched, with constant temperature, to be used in cnSearchSA
`maxIter`	an integer, the total number of iterations, thus orders, to be processed, to be used in cnSearchSA
`filename`	name of the sif file saved, default BAYESIAN

Details

This function transforms the data in a format compatible with catnet package, infers the network using the Stochastic Network Search as implemented in catnet (see cnSearchSA), computes the consensus model of the models returned by cnSearchSA considering only links that have a frequency of appearence greater than 0.1 and returns the model in the sif format.

Value

`sif`	the inferred data-driven network in sif format

Author(s)

F.Eduati

See Also

MapDDN2Model

Examples

```
library(CellNOptR)
data(CNOlistDREAM,package="CellNOptR")
DDN<-Binference(CNOlistDREAM, tempCheckOrders=10, maxIter=100,
                filename="BAYESIAN")
```

MIinference *Mutual information based network inference*

Description

This function uses data (CNOlist) to infer a data-driven network using the mutual information based appoaches ARACNe and CLR as implemented in the minet package.

Usage

```
MIinference(CNOlist, method="ARACNE", PKNgraph=NULL,
            filename="ARACNE")
```

Arguments

CNOlist	a CNOlist structure, as produced by makeCNOlist
method	a character, the name of the method to be used: ARACNE or CLR. Default, ARACNE
PKNgraph	a network to be used for comparison to assess the directionality of some links. Default is NULL.
filename	name of the sif file saved, default ARACNE

Details

This function transforms the data in a format compatible with minet package, infers the network using aracne or clr as implemented in the minet package and returns the network in the sif format. It is important to notice that mutual information approaches do not allow for determining the directionality of the links thus both directions are considered. The function allows to give as input a network in graph format (graph package, see sif2graph to convert from sif to graph format) to be used as comparison to assess the directionality of some links, e.g. PKN.

Value

sif	the inferred data-driven network in sif format

Author(s)

F.Eduati

References

P. E. Meyer, F. Lafitte and G. Bontempi (2008). MINET: An open source R/Bioconductor Package for Mutual Information based Network Inference. Bioinformatics, Vol 9, 2008

See Also

MapDDN2Model, sif2graph, model2sif

Examples

```
library(CellNOptR)
data(CNOlistDREAM,package="CellNOptR")
data(DreamModel,package="CellNOptR")
PKNgraph<-sif2graph(model2sif(DreamModel))

method="ARACNE"
#method="CLR"
DDN<-MIinference(CNOlist=CNOlistDREAM, method=method,
                 PKNgraph=PKNgraph, filename=method)
```

MapBTables2Model *Integrate Boolean tables with the model*

Description

This function infers the network from the Boolean tables and integrates it with the network encoded in the model (generally derived from prior knowledge), adding links that are missing.

Usage

```
MapBTables2Model(BTable,Model,optimRes=NA,allInter=TRUE)
```

Arguments

BTable	a BTable list, as created by makeBTables
Model	a Model list, as created by readSif
optimRes	a bit string with the reaction of the model to be considered, default considers all reactions
allInter	one new link in the network can correspond to more links in the model, set it to TRUE if you want to add all possible links, FALSE to add only one link, default is TRUE

Details

The function receive as input the Boolean Tables, infers the data-driven network form them (as descibed in (Eduati et al., PLoS ONE, 2010)) and integrates it with the model, returning a new model with the integrated links. If the Model is not given as input (Model=NULL), the data-driven network is returned as model.

Value

a new Model with the integrated links and an additional field:

`indexIntegr` a vector with the indexes of the integrated links

Author(s)

F.Eduati

References

F. Eduati, A. Corradin, B. Di Camillo, G. Toffolo. A Boolean approach to linear prediction for signaling network modeling. PLoS ONE; 5(9): e12789.

See Also

readSif, readMIDAS, makeBTables

Examples

```
library(CellNOptR)
data(CNOlistDREAM,package="CellNOptR")
data(DreamModel,package="CellNOptR")
res<-preprocessing(Data=CNOlistDREAM, Model=DreamModel)
Model<-res$model
BTable <- makeBTables(CNOlist=CNOlistDREAM, k=2, measErr=c(0.1, 0))
modelIntegr <- MapBTables2Model(BTable=BTable,Model=Model,allInter=TRUE)
# modelIntegr$reacID[modelIntegr$indexIntegr] to see the integrated links
```

MapDDN2Model *Integrate data-drive network with the model*

Description

This function integrates the data-driven network (in sif format) with the network encoded in the model (generally derived from prior knowledge), adding links that are missing.

Usage

```
MapDDN2Model(DDN,Model,CNOlist,allInter=TRUE)
```

Arguments

DDN	a sif file encoding a data-driven network, as created by Binference or MInference
Model	a Model list, as created by readSif
CNOlist	a CNOlist, as created by makeCNOlist
allInter	one new link in the network can correspond to more links in the model, set it to TRUE if you want to add all possible links, FALSE to add only one link, default is TRUE

Details

The function receives as input a sif file with the data-driven network, as created by Binference or MInference, and integrates it with the model, returning a new model with the integrated links.

Value

a new Model with the integrated links and an additional field:

indexIntegr	a vector with the indexes of the integrated links

Author(s)

F.Eduati

See Also

readSif, readMIDAS, Binference, MInference

Examples

```
library(CellNOptR)
data(CNOlistDREAM,package="CellNOptR")
data(DreamModel,package="CellNOptR")
res<-preprocessing(Data=CNOlistDREAM, Model=DreamModel)
Model<-res$model

DDN<-Binference(CNOlistDREAM, tempCheckOrders=10, maxIter=100,
                filename="BAYESIAN")

modelIntegr<-MapDDN2Model(DDN=DDN,Model=Model,CNOlist=CNOlistDREAM)
```

PPIweight *Weight links using protien-protein interactions*

Description

This function weights links integrated in the model using information derived from protein-protein interaction networks (PINs).

Usage

```
PPIweight(modelIntegr,PKNmodel,CNOlist,UniprotID,PPINigraph)
```

Arguments

`modelIntegr`	the integrated model as created by MapDDN2Model or MapBTables2Model
`PKNmodel`	the model of the original prior-knowledge network
`CNOlist`	a CNOlisi, as created by makeCNOlist
`UniprotID`	a list with the Uniprot identifiers of proteins in the PKN
`PPINigraph`	the igraph (igraph package) of the PIN

Details

The basic idea is that if, for a directed link A <e2><86><92> B integrated in the PKN, there is a corresponding path in the PIN, it is more plausible that there is a molecular pathway A <e2><86><92> B. Because shorter paths are more feasible, as a first approximation the shortest path length between A and B in the PIN can be used as a reliability score for the integrated link. Since the optimization is performed on a compressed version of the PKN, one link integrated in the compressed network generally corresponds to multiple possible links integrated in the PKN and the shortes path of all. The weight for each integrated link in the compressed network is thus computed as (1 + the inverse of the sum of the inverse of the corresponding PKN of the shortest paths in the PIN). A high quality network of known human physical protein-protein interaction assembled from multiple databases is provided with the package: interactions were included only if validated by at least one <e2><80><98>binary<e2><80><99> experimental method in a published paper and the number of experimental evidences was reported for each interaction.

Value

This function returns a list with elements:

`modelIntegr`	the input modelIntegr with an additional field: a vector with the weights of the integrated links
`PPINigraph`	the input PPINigraph with added colour attributes for edges and nodes
`ListPaths`	the list with, for each integrated link, the list of the shortes paths in the PIN corresponding to the paths in the PKN
`saveShortestPath`	has the same structure of ListPaths but contains only the length of the paths in terms of number of edges

Author(s)

F.Eduati

See Also

MapDDN2Model, MapBTables2Model, gaBinaryT1int

Examples

```
library(CellNOptR)
data(CNOlistDREAM,package="CellNOptR")
data(DreamModel,package="CellNOptR")
data(UniprotIDdream,package="CNORfeeder")
data(PPINigraph,package="CNORfeeder")
res<-preprocessing(Data=CNOlistDREAM, Model=DreamModel)
Model<-res$model

BTable <- makeBTables(CNOlist=CNOlistDREAM, k=2, measErr=c(0.1, 0))
modelIntegr <- MapBTables2Model(BTable=BTable,Model=Model,allInter=TRUE)

# the followig step may take a while
## Not run:
resPPIweight <- PPIweight(modelIntegr=modelIntegr,PKNmodel=DreamModel,
                CNOlist=CNOlistDREAM,UniprotID=UniprotIDdream,
                PPINigraph=PPINigraph)
## End(Not run)
```

`gaBinaryT1int` *Genetic algorithm used to optimise a model*

Description

This function is the genetic algorithm to be used to optimise a model by fitting to data containing one time point. It is the function gaBinaryT1 of CellNOptR modified in orter to differently weights for the integrated links

Usage

```
gaBinaryT1int(CNOlist, Model, SimList, indexList, sizeFac = 1e-04,
              integrFac=10, NAFac = 1, initBstring, PopSize = 50,
              Pmutation = 0.5, MaxTime = 60, maxGens = 500,
              StallGenMax = 100, SelPress = 1.2, elitism = 5,
              RelTol = 0.1, verbose=TRUE)
```

Arguments

`CNOlist`	a CNOlist on which the score is based (based on valueSignals[[2]], i.e. data at t1)
`Model`	a Model list
`SimList`	a list that contains additional fields for the simulator, as created by prep4Sim applied to the model above
`indexList`	a list of indexes of species stimulated/inhibited/signals, as produced by indexfinder applied on the model and CNOlist above
`sizeFac`	the scaling factor for the size term in the objective function, default to 0.0001
`integrFac`	the scaling factor for the integration term in the objective function, default to 10

NAFac	the scaling factor for the NA term in the objective function, default to 1
initBstring	an initial bitsring to be tested, should be of the same size as the number of reactions in the model above
PopSize	the population size for the genetic algorithm, default set to 50
Pmutation	the mutation probability for the genetic algorithm, default set to 0.5
MaxTime	the maximum optimisation time in seconds, default set to 60
maxGens	the maximum number of generations in the genetic algorithm, default set to 500
StallGenMax	the maximum number of stall generations in the genetic algorithm, default to 100
SelPress	the selective pressure in the genetic algorithm, default set to 1.2
elitism	the number of best individuals that are propagated to the next generation in the genetic algorithm, default set to 5
RelTol	the relative tolerance for the best bitstring reported by the genetic algorithm, i.e., how different from the best solution, default set to 0.1
verbose	logical (default to TRUE) do you want the statistics of each generation to be printed on the screen?

Details

The whole procedure is described in details in Saez-Rodriguez et al. (2009), see gaBinaryT1 for detaied description. The only additional input is integrFac, that is used to penalize more links inferred strictly from data using reverse-engineering approaches and integrated using MapBTables2Model or MapDDN2Model. If the input model has the field IntegrPPIscores, as given by PPIweight, this is used to differently penalize integrated links in the computation of the score for each element of the population.

Value

This function returns a list with elements:

bString	the best bitstring
Results	a matrix with columns "Generation", "Best_score", "Best_bitString", "Stall_Generation", "Avg_Score_Gen", "Best_score_Gen", "Best_bit_Gen", "Iter_time"
StringsTol	the bitstrings whose scores are within the tolerance
StringsTolScores	the scores of the above-mentioned strings

Author(s)

F.Eduati

References

J. Saez-Rodriguez, L. G. Alexopoulos, J. Epperlein, R. Samaga, D. A. Lauffenburger, S. Klamt and P. K. Sorger. Discrete logic modeling as a means to link protein signaling networks with functional analysis of mammalian signal transduction, Molecular Systems Biology, 5:331, 2009.

See Also

prep4Sim, indexFinder, MapDDN2Model, MapBTables2Model PPIweight

Examples

```
library(CellNOptR)
data(CNOlistDREAM,package="CellNOptR")
data(DreamModel,package="CellNOptR")
data(UniprotIDdream,package="CNORfeeder")
data(PPINigraph,package="CNORfeeder")
res<-preprocessing(Data=CNOlistDREAM, Model=DreamModel)
Model <- res$model

BTable <- makeBTables(CNOlist=CNOlistDREAM, k=2, measErr=c(0.1, 0))
modelIntegr <- MapBTables2Model(BTable=BTable,Model=Model,allInter=TRUE)

# the followig step may take a while
## Not run:
resPPIweight <- PPIweight(modelIntegr=modelIntegr,PKNmodel=DreamModel,
                          CNOlist=CNOlistDREAM,UniprotID=UniprotIDdream,
                          PPINigraph=PPINigraph)
modelIntegr <- resPPIweight$modelIntegr
## End(Not run)

DreamFields4Sim <- prep4Sim(modelIntegr)
initBstring <- rep(1,length(modelIntegr$reacID))
DreamT1opt <- gaBinaryT1int(
CNOlist=CNOlistDREAM,
Model=modelIntegr,
SimList=DreamFields4Sim,
indexList=res$indices,
initBstring=initBstring,
maxGens=2,
PopSize=5,
verbose=FALSE)
```

`makeBTables` *Make Boolean tables*

Description

This function uses data (CNOlist) to infer a Boolean table for each measured protein, codifying if a particular stimulus inhibitor combination affects the protein. A stimulus or an inhibitor significantly affects an output protein if it is able to modify its activity level of a quantity that exceeds the uncertainty associated with its measurement.

Usage

```
makeBTables(CNOlist, k=2, measErr=c(0.1, 0), timePoint=NA)
```

Arguments

`CNOlist`	a CNOlist structure, as produced by makeCNOlist
`k`	a parameter that determine the threshold of significancy of the effect of stimuli and inhibitors, default to 2
`measErr`	a 2 value vector (err1, err2) defining the error model of the data as sd^2 = err1^2 + (err2*data)^2, default to c(0.1, 0)

`timePoint` the time point to be considered for the inference of the Boolean tables, if not specified all time points are consideres

Details

This function computes the fist step of FEED to reverse engineer the network strictly from data, i.e. the inference of Boolean tables, as described in (Eduati et al., PLoS ONE, 2010). For each protein, a Boolean table is inferred having one columns for each stimulus and one row for each inhibitor. If a stimulus produces a significant effect on the activity level of the protein this is codified with a 1 in the corresponding column, if also the inhibitor affect the protein there is a 2 in the corresponding cell. The sign of the regulation is coded in separate tables.

Value

this function returns a list with fields:

`namesSignals` a vector of names of signals

`tables` a list with one Boolean table for each protein codifying the effect of stimuli (columns) and inhibitors (rows), 1 if the stimulus affect the protein, 2 if also the inhibior does

`NotMatStim` has the same format as tables but just contains a 1 if the regulation has a negative effect, and 0 otherwise

`NotMatInhib` has the same format as tables but just contains a 1 if the regulation has a negative effect, and 0 otherwise

Author(s)

F.Eduati

References

F. Eduati, A. Corradin, B. Di Camillo, G. Toffolo. A Boolean approach to linear prediction for signaling network modeling. PLoS ONE; 5(9): e12789.

See Also

makeCNOlist, MapBTables2Model

Examples

```
library(CellNOptR)
data(CNOlistDREAM,package="CellNOptR")
BTable <- makeBTables(CNOlist=CNOlistDREAM, k=2, measErr=c(0.1, 0))
```

mode2sif *Convert model to sif*

Description

This function converts a network form model format to sif format and saves the sif file.

Usage

```
mode2sif(Model,optimRes=NA,writeSif=FALSE, filename="Model")
```

Arguments

Model	the model, as created by reasSif
optimRes	a bit string with the reaction of the model to be considered, default considers all reactions
writeSif	if you want to save the sif file set it to TRUE, otherwise FALSE. Default is FALSE
filename	name of the sif file. Default is Model

Details

This function takes as input, the model and the bit string and saves the corresponding model in sif format.

Value

sifFile	the corresponding sif

Author(s)

F.Eduati

See Also

sif2graph

sif2graph *Convert sif to graph*

Description

This function converts a network form sif format to graph format.

Usage

```
sif2graph(sif)
```

Arguments

sif	the name of a sif file

Details

This function takes in a single argument, sifFile, that points to a previous knowledge network in .sif format i.e. sourceNode-tab-sign-tab-targetNode. If there are ANDs they should be introduced as dummy nodes called and# (don't forget the number after "and" otherwise this won't be recognised). Please be aware that "and" nodes are not expected to be negated, i.e. there are not supposed to be !and1=xyz because that amounts to inverting the sign of all inputs of and1, which is more simply done at the inputs level.

Value

g	the corresponding graph

Author(s)

F.Eduati

See Also

model2sif

Index