Package 'lpNet'

April 10, 2015

Type Package

Title Linear Programming Model for Network Inference

Version 1.6.0

Date 2013-01-11

Author Bettina Knapp, Johanna Mazur, Lars Kaderali

Maintainer Bettina Knapp <bettina.knapp@tu-dresden.de>

Depends lpSolve, nem

Description lpNet takes perturbation data as input and generates an LP model which allows the inference of signaling networks. For parameter identification either leave-one-out cross-validation or stratified n-fold cross-validation can be used.

License Artistic License 2.0

biocViews Network

R topics documented:

lpNet-package *Network Inference Of Perturbation Data Using a Linear Programming Approach.*

Description

lpNet takes perturbation data as input and generates an LP model which allows the inference of signaling networks. For parameter identification either leave-one-out cross-validation or stratified n-fold cross-validation can be used.

Details

Package:	lpNet
Type:	Package
Version:	1.0
Date:	2013-01-11
License:	Artistic License 2.0

References

B. Knapp, RNA Interference Data: from a Statistical Analysis to Network Inference, PhD Thesis, 2012, http://archiv.ub.uni-heidelberg.de/volltextserver/13322/

calcActivation *Calculate Activation Matrix*

Description

Calculate the activation matrix assuming that the signaling is deterministically propagated along the network. For a given network and given perturbation experiments the theoretical states of the genes are computed. So if a gene has been silenced in an experiment, then the state of this gene is assumed to be inactive, otherwise it is active, if its inflow (coming from parent nodes) is activating. Cycles within a network are not resolved, therefore this function can be used only for networks without cycles.

Usage

```
calcActivation(T_nw, b, n, K)
```

Arguments

T_nw	Adjacency matrix: the network which is used to compute the activities and inactivities.
b	Vector of 0/1 values describing the experiments (entry is 0 if gene is inactivated in the respetive experiment and 1 otherwise). The measurements of the genes of each experiment are appended as a long vector.
n	Integer: number of genes.
K	Integer: number of pertrubations experiments.

Value

Matrix of 0/1 values; rows corresponding to genes, columns to experiments. If an entry is 1, it means that the corresponding gene is active in the corresponing experiment and inactive otherwise.

Examples

```
n <- 5 # number of genes
K <- 7 # number of perturbations experiments

# perturbation vector, entry is 0 if gene is inactivated and 1 otherwise
b <- c(0,1,1,1,1, # perturbation exp1: gene 1 perturbed, gene 2-5 unperturbed
1,0,1,1,1,        # perturbation exp2: gene 2 perturbed, gene 1,3,4,5 unperturbed
1,1,0,1,1,        # perturbation exp3....
1,1,1,0,1,
1,1,1,1,0,
1,0,0,1,1,
1,1,1,1,1)
# example network
T_nw <- matrix(c(0,1,1,0,0,0,0,0,-1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0),
  nrow=n,ncol=n,byrow=TRUE)
# compute theoretical activation of genes from example network with given perturbations
act_mat <- calcActivation(T_nw,b,n,K)
```

---

calcPrediction *Calculate Predicted Observation.*

Description

Calculate the predicted observation of a perturbation experiment. If observations of an experiment are missing this function can be used to determine for a given network the predicted outcome. The missing measurement is predicted from two normal distributions, one for observations coming from active and one coming from inactive genes.

Usage

```
calcPredictionLOOCV(kds, adja, obs, delta, rem_kd, rem_gene,
  active_mu, active_sd, inactive_mu, inactive_sd)
calcPredictionKfoldCV(adja, b, n, K,
  active_mu, active_sd, inactive_mu, inactive_sd)
```

Arguments

kds	Matrix of 0/1 values representing the perturbation experiments (entry is 0 if gene is inactivated in the respective experiment and 1 otherwise). One column corresponds to one experiment and each row represents a gene.
adja	Numeric matrix: the adjacency matrix of the given network.
obs	Numeric matrix: the observation matrix.
delta	Numeric vector defining the thresholds for each gene to determine its observation to be active or inactive.
rem_kd	Integer: the index of the experiment which is missing.
rem_gene	Integer: the index of the gene which is missing. Together with rem_kd, the two indices correspond to the observation of the gene rem_gene at experiment rem_kd.
active_mu	Numeric: the average value assumed for observations coming from activated nodes. The parameter active_mu and active_sd are used for predicting the observations of the normal distribution of activate genes.
active_sd	Numeric: the variation assumed for observations coming from activated nodes. The parameter active_mu and active_sd are used for predicting the observations of the normal distribution of activate genes.
inactive_mu	Numeric: the average value assumed for observations coming from inactivated nodes. The parameter inactive_mu and inactive_sd are used for predicting the observations of the normal distribution of inactivate genes.
inactive_sd	Numeric: the variation assumed for observations coming from inactivated nodes. The parameter inactive_mu and inactive_sd are used for predicting the observations of the normal distribution of inactivate genes.
b	Vector of 0/1 values describing the experiments (entry is 0 if gene is inactivated in the respetive experiment and 1 otherwise). The measurements of the genes of each experiment are appended as a long vector.
n	Integer: number of genes.
K	Integer: number of perturbations experiments.

Value

Numeric: the predicted observation(s).

See Also

kfoldCV, loocv

Examples

```
n <- 5 # number of genes
K <- 7 # number of experiments

# perturbation vector, entry is 0 if gene is inactivated and 1 otherwise
b <- c(0,1,1,1,1, # perturbation exp1: gene 1 perturbed, gene 2-5 unperturbed
```

```
1,0,1,1,1,        # perturbation exp2: gene 2 perturbed, gene 1,3,4,5 unperturbed
1,1,0,1,1,        # perturbation exp3....
1,1,1,0,1,
1,1,1,1,0,
1,0,0,1,1,
1,1,1,1,1)
T_nw <- matrix(c(0,1,1,0,0,0,0,NA,-1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0),
  nrow=n,ncol=n,byrow=TRUE)

# define the parameters for the observation generated from the normal distributions
active_mu <- 0.9
inactive_mu <- 0.5
active_sd <- inactive_sd <- 0.01

# compute the predicted observation matrix for the "kfoldCV"
calcPredictionKfoldCV(T_nw, b, n, K, active_mu, active_sd, inactive_mu, inactive_sd)

#### LOOCV
# perturbation experiments written as a matrix
kds <- matrix(b,nrow=n,ncol=K)

# generate random observation matrix
obs <- matrix(rnorm(35),nrow=n,ncol=K)
# define delta
delta <- rep(0.75,5)
# define the observation to be removed
rem_kd <- 3
rem_gene <- 2
obs[rem_gene,rem_kd] <- NA
# compute the predicted value
calcPredictionLOOCV(kds, T_nw, obs, delta, rem_kd, rem_gene,
  active_mu, active_sd, inactive_mu, inactive_sd)
```

`calcRangeLambda` *Compute Range Of Penalty Parameter Lambda.*

Description

The penalty parameter lambda can range from zero to infinity and it controls the introduction of slack variables in the network inference lp model. To limit the introduction of slack variables we restrict lambda to be not larger than lambdaMax (=the number of slack variables times the variance of all measurements given). This function computes the range from zero to lambdaMax with a given stepsize.

Usage

```
calcRangeLambda(delta, obs, stepsize)
```

Arguments

delta	Numeric vector defining the thresholds for each gene to determine its observation to be active or inactive.
obs	Numeric matrix of the observed measurements.
stepsize	Numeric: giving the step size.

Value

Numeric vector of possible values for lambda.

Examples

```
# generate random observation matrix with 5 experiments and 5 genes
obs <- matrix(rnorm(5*5,1,0.1),nrow=5,ncol=5)
# define delta to be 1 for each gene
delta <- rep(1,5)
lambda <- calcRangeLambda(delta,obs,stepsize=0.01)
```

CV *Cross-validation*

Description

Performs a stratified k-fold cross-validation or a Leave-One-Out cross-validation.

Usage

```
kfoldCV(times, obs, n, b, K, delta, lambda, annot, annot_node, kfold,
  active_mu, active_sd, inactive_mu, inactive_sd,
  prior = NULL, sourceNode = NULL, sinkNode = NULL, allint = FALSE, allpos = FALSE)
loocv(times, obs, n, b, K, delta, lambda, annot, annot_node,
  active_mu, active_sd, inactive_mu, inactive_sd,
  prior = NULL, sourceNode = NULL, sinkNode = NULL, allint = FALSE, allpos = FALSE)
```

Arguments

times	Integer: the number of times the cross-validation shall be performed.
obs	Numeric matrix: the measured observations.
n	Integer: number of genes.
b	Vector of 0/1 values describing the experiments (entry is 0 if gene is inactivated in the respetive experiment and 1 otherwise). The measurements of the genes of each experiment are appended as a long vector.
K	Integer: number of perturbation experiments.
delta	Numeric vector defining the thresholds to determine an observation active or inactive.

lambda	Numeric value defining the penalty parameter lambda. It can range from zero to infinity and it controls the introduction of slack variables in the network inference lp model.
annot	Vector of character strings: the annotation of the edges as returned by "getEdgeAnnot".
annot_node	Vector of character strings: the annoation of the nodes.
kfold	Integer value of the number "k" in the k-fold cross-calidation.
active_mu	Numeric: the average value assumed for observations coming from activated nodes. The parameter active_mu and active_sd are used for predicting the observations of the normal distribution of activate genes.
active_sd	Numeric: the variation assumed for observations coming from activated nodes. The parameter active_mu and active_sd are used for predicting the observations of the normal distribution of activate genes.
inactive_mu	Numeric: the average value assumed for observations coming from inactivated nodes. The parameter inactive_mu and inactive_sd are used for predicting the observations of the normal distribution of inactivate genes.
inactive_sd	Numeric: the variation assumed for observations coming from inactivated nodes. The parameter inactive_mu and inactive_sd are used for predicting the observations of the normal distribution of inactivate genes.
prior	Prior knowledge, given as a list of constraints. Each constraint consists of a vector with four entries describing the prior knowledge of one edge. For example the edge between node 1 and 2, called w+_1_2, is defined to be bigger than 1 with constraint c("w+_1_2",1,">",2). The first entry specifies the annotation of the edge (see function "getEdgeAnnot") and the second defines the coefficient of the objective function (see parameter "objective.in" in the "lp" function of the package "lpSolve"). Furthermore, the third, respectively the fourth elements give the direction, respectively the right-hand side of the constraint (see the parameters "const.dir", respectively "const.rhs" in the "lp" function of the package "lpSolve").
sourceNode	Integer vector: indices of the known source nodes.
sinkNode	Integer vector: indices of the known sink nodes.
allint	Logical: should all variables be integer? Corresponds to an Integer Linear Program (see "lp" function in package "lpSolve"). Default: FALSE.
allpos	Logical: should all variables be positive? Corresponds to learning only activating edges. Default: FALSE.

Value

A list of

MSE	The mean squared error (MSE) of predicted and observed measurements of the corresponding cross-validation step.
edges_all	The learned edge weights for each cross-validation step.

Examples

```
annot_node <- seq(1,5)
n <- 5 # number of genes
K <- 7 # number of perturbation experiments

annot <- getEdgeAnnot(n)

# activation (knockdown) vector (entry is 0 if gene is inactivated in the respetive experiment)
b <- c(0,1,1,1,1,
1,0,1,1,1,
1,1,0,1,1,
1,1,1,0,1,
1,1,1,1,0,
1,0,0,1,1,
1,1,1,1,1)

T_nw <- matrix(c(0,1,1,0,0,0,0,0,-1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0),nrow=n,ncol=n,byrow=TRUE)
colnames(T_nw) <- rownames(T_nw) <- annot_node

## calculate observation matrix with given parameters for the
# Gaussian distributions for activation and deactivation
active_mu <- 0.95
inactive_mu <- 0.56
active_sd <- 0.1
inactive_sd <- 0.1
# use three replicates
replnum <- 3

# generation of random observation matrix
obs <- matrix(rnorm(35),nrow=n,ncol=K)

times <- kfold <- 10   # can be increased i.e. to 1000 to produce stable results

# define delta
delta <- apply(obs,1,mean,na.rm=TRUE)

#### LOOCV
loocv_res <- loocv(times,obs,n,b,K,delta,lambda=1,annot,annot_node,
  active_mu,active_sd,inactive_mu,inactive_sd)

#### K-fold CV
kcv_res <- kfoldCV(times,obs,n,b,K,delta,lambda=1,annot,annot_node,kfold,
  active_mu,active_sd,inactive_mu,inactive_sd)
```

---

doILP *Do The Network Inference With The Linear Programming Approach.*

Description

This function converts observation data into a linear programming problem.

Usage

```
doILP(obs, delta, lambda, b, n, K, annot,
  prior = NULL, sourceNode=NULL, sinkNode=NULL, all.int = FALSE, all.pos = FALSE)
```

Arguments

obs	Numeric matrix: the given observations.
delta	Numeric vector defining the thresholds for each gene to determine its observation to be active or inactive.
lambda	Numeric value defining the penalty parameter lambda. It can range from zero to infinity and it controls the introduction of slack variables in the network inference lp model.
b	Vector of 0/1 values describing the experiments (entry is 0 if gene is inactivated in the respetive experiment and 1 otherwise). The measurements of the genes of each experiment are appended as a long vector.
n	Integer: number of genes.
K	Integer: number of perturbation experiments.
annot	Vector of character strings: the annotation of the edges as returned by "getEdgeAnnot".
prior	Prior knowledge, given as a list of constraints. Each constraint consists of a vector with four entries describing the prior knowledge of one edge. For example the edge between node 1 and 2, called w+_1_2, is defined to be bigger than 1 with constraint c("w+_1_2",1,">",2). The first entry specifies the annotation of the edge (see function "getEdgeAnnot") and the second defines the coefficient of the objective function (see parameter "objective.in" in the "lp" function of the package "lpSolve"). Furthermore, the third, respectively the fourth elements give the direction, respectively the right-hand side of the constraint (see the parameters "const.dir", respectively "const.rhs" in the "lp" function of the package "lpSolve").
sourceNode	Integer vector: indices of the known source nodes.
sinkNode	Integer vector: indices of the known sink nodes.
all.int	Logical: should all variables be integer? Corresponds to an Integer Linear Program (see "lp" function in package "lpSolve"). Default: FALSE.
all.pos	Logical: should all variables be positive? Corresponds to learning only activating edges. Default: FALSE.

Value

An lp object. See "lp.object" in package "lpSolve" for details.

Examples

```
n <- 5 # number of genes
K <- 7 # number of perturbation experiments
annot <- getEdgeAnnot(n)
```

```
# generation of random observation matrix
obs <- matrix(rnorm(35),nrow=n,ncol=K)
# define delta
delta <- apply(obs,1,mean,na.rm=TRUE)

# activation (knockdown) vector (entry is 0 if gene is inactivated in the respetive experiment)
b <- c(0,1,1,1,1,
1,0,1,1,1,
1,1,0,1,1,
1,1,1,0,1,
1,1,1,1,0,
1,0,0,1,1,
1,1,1,1,1)

res <- doILP(obs,delta,lambda=1,b,n,K,annot)
```

---

getAdja *Get Adjacency Matrix.*

Description

The function returns the adjacency matrix of the network computed with the "doILP" function.

Usage

```
getAdja(result, numnodes)
```

Arguments

result	Result returned by the "doILP" function.
numnodes	Integer: the number of nodes of the inferred network.

Value

Numeric matrix: the adjacency matrix of the network.

See Also

doILP

Examples

```
# generate random observation matrix with 5 experiments and 5 genes
obs <- matrix(rnorm(5*5,1,0.1),nrow=5,ncol=5)
# define the perturbations
b <- c(0,1,1,1,1,
1,0,1,1,1,
1,1,0,1,1,
```

```
1,1,1,0,1)
n <- 5 # number of genes
K <- 4 # number of knockdowns
# annotation of the edges
annot <- getEdgeAnnot(n)
# define delta
delta <- rep(1,5)
# infer the network
res <- doILP(obs,delta,lambda=1,b,n,K,annot)

# make the adjacency matrix
adja <- getAdja(res,n)
```

`getEdgeAnnot` *Get the annotation of the edges.*

Description

The function returns the annotation of the edges needed for the LP. Positive edges are annotated with "w+" and negative with "w-". The given nodes are just enumerated from 1 to n and the edge between node i and j is given by "w+_i_j" for the positive, respectively by "w-_i_j" for the negative edges. The annotation "w_i_^_0" defines the baseline activity of gene i.

Usage

```
getEdgeAnnot(n,allpos)
```

Arguments

`n`	Integer: number of genes.
`allpos`	Logical: should all edges be positive? Corresponds to learning only activating edges. Default: FALSE.

Value

Vector of character strings.

Examples

```
n <- 5
annot <- getEdgeAnnot(n)
```

getObsMat *Get Observation Matrix.*

Description

The function generates from the activation matrix, computed with "calcActivation", the observation matrix where active/inactive observations are generated from a normal distribution with the average and variation as given in the parameters.

Usage

```
getObsMat(act_mat, active_mu, active_sd, inactive_mu, inactive_sd)
```

Arguments

act_mat	Matrix of 0/1 values called the activation matrix. Rows correspond to genes, columns to experiments. If an entry is 1, it means that the corresponding gene is active in the corresponing experiment and inactive otherwise.
active_mu	Numeric: the average value assumed for observations coming from activated nodes. The parameter active_mu and active_sd are used for predicting the observations of the normal distribution of activate genes.
active_sd	Numeric: the variation assumed for observations coming from activated nodes. The parameter active_mu and active_sd are used for predicting the observations of the normal distribution of activate genes.
inactive_mu	Numeric: the average value assumed for observations coming from inactivated nodes. The parameter inactive_mu and inactive_sd are used for predicting the observations of the normal distribution of inactivate genes.
inactive_sd	Numeric: the variation assumed for observations coming from inactivated nodes. The parameter inactive_mu and inactive_sd are used for predicting the observations of the normal distribution of inactivate genes.

Value

Numeric matrix: the observation matrix

See Also

calcActivation

Examples

```
n <- 5 # number of genes
K <- 7 # number of knockdowns

# perturbation vector, entry is 0 if gene is inactivated and 1 otherwise
b <- c(0,1,1,1,1, # perturbation exp1: gene 1 perturbed, gene 2-5 unperturbed
1,0,1,1,1,        # perturbation exp2: gene 2 perturbed, gene 1,3,4,5 unperturbed
```

```
1,1,0,1,1,        # perturbation exp3....
1,1,1,0,1,
1,1,1,1,0,
1,0,0,1,1,
1,1,1,1,1)
T_nw <- matrix(c(0,1,1,0,0,0,0,0,-1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0),nrow=n,ncol=n,byrow=TRUE)
act_mat <- calcActivation(T_nw,b,n,K)

# define the parameters for the observation generated from the normal distribution
active_mu <- 0.9
inactive_mu <- 0.5
active_sd <- inactive_sd <- 0.1

# compute the observations matrix
getObsMat(act_mat, active_mu, active_sd, inactive_mu, inactive_sd)
```

`getSampleAdja` *Get The Sample Adjacency.*

Description

The function computes the adjacency of the edges computed in each step of the "loocv" or the "kfoldCV" function. If the variance of each edge shall be taken into account use "getSampleAdjaMAD", otherwise "getSampleAdja".

Usage

```
getSampleAdjaMAD(edges_all, numnodes, annot_node,
  method = median, method2 = mad, septype = "->")
getSampleAdja(edges_all, numnodes, annot_node, method = median, septype = "->")
```

Arguments

`edges_all`	The inferred edges using the "loocv" or the "kfoldCV" function.
`numnodes`	Integer: the number of nodes.
`annot_node`	Vector of character strings: the annoation of the nodes.
`method`	Character string: the method used to summarize the edges of the individual steps. Default: "median".
`method2`	Character string: the method used for the computation of the variation of the edges of the individual steps. Default: "mad".
`septype`	Character string: the type of separation of two nodes in the annot string vector. Default: "->".

Value

Numeric matrix: the adjacency matrix.

See Also

loocv, kfoldCV

Examples

```
# compute random edge weights
edges_all <- matrix(rnorm(5*6),nrow=5,ncol=6)
# annotation of the edges as returned by "loocv" and kfoldCV
colnames(edges_all) <- c("1->2","1->3","2->1","2->3","3->1","3->2")
# annotation of the nodes
annot_node <- c(1,2,3)
getSampleAdjaMAD(edges_all, numnodes=3, annot_node, method = "median",
  method2 = "mad", septype = "->")
getSampleAdja(edges_all, numnodes=3, annot_node, method = "median", septype = "->")
```

---

summarizeRepl *Summarize Replicate Measurements*

Description

The function returns the the summarized replicate measuremenst.

Usage

```
summarizeRepl(data,type=median)
```

Arguments

data	The data matrix.
type	The summarization type which shall be used. Default: median.

Value

Numeric matrix: the summarized data.

Examples

```
data("SahinRNAi2008")
## process data
dataStim <- dat.normalized[dat.normalized[,17]==1,-17]

# summarize replicates
dataSt <- t(summarizeRepl(dataStim,type=mean))
```

Index