Package ‘predictionet’

April 15, 2017

Type Package
Title Inference for predictive networks designed for (but not limited to) genomic data
Version 1.20.0
Date 2011-10-05
Author Benjamin Haibe-Kains, Catharina Olsen, Gianluca Bontempi, John Quackenbush
Maintainer Benjamin Haibe-Kains <bhaibeka@jimmy.harvard.edu>, Catharina Olsen <colsen@ulb.ac.be>

Description This package contains a set of functions related to network inference combining genomic data and prior information extracted from biomedical literature and structured biological databases. The main function is able to generate networks using Bayesian or regression-based inference methods; while the former is limited to < 100 of variables, the latter may infer networks with hundreds of variables. Several statistics at the edge and node levels have been implemented (edge stability, predictive ability of each node, ...) in order to help the user to focus on high quality subnetworks. Ultimately, this package is used in the 'Predictive Networks' web application developed by the Dana-Farber Cancer Institute in collaboration with Entagen.

Depends igraph, catnet
Suggests network, minet, knitr
Imports penalized, RBGL, MASS
License Artistic-2.0

LazyData yes
biocViews GraphAndNetwork, NetworkInference
NeedsCompilation yes

R topics documented:
predictionet-package ................................................................. 2
adj.get.hops ................................................................. 3
Inference for predictive networks designed for (but not limited to) genomic data

This package contains a set of functions related to network inference combining genomic data and prior information extracted from biomedical literature and structured biological databases. The main function is able to generate networks using bayesian or regression-based inference methods; while the former is limited to < 100 of variables, the latter may infer network with hundreds of variables. Several statistics at the edge and node levels have been implemented (edge stability, predictive ability of each node, ...) in order to help the user to focus on high quality subnetworks. Ultimately, this package is used in the 'Predictive Networks' web application developed by the Dana-Farber Cancer Institute in collaboration with

Details

Package: predictionet
Type: Package
Version: 1.20.0
Date: 2011-10-05
License: Artistic 2.0
LazyLoad: yes

Author(s)

Benjamin Haibe-Kains, Catharina Olsen, Gianluca Bontempi, John Quackenbush
- Computational Biology and Functional Genomics, Dana-Farber Cancer Institute, Boston, MA, USA

http://compbio.dfci.harvard.edu/
adj.get.hops

- Center for Cancer Computational Biology, Dana-Farber Cancer Institute, Boston, MA, USA
http://cccb.dfci.harvard.edu/index.html
- Machine Learning Group (MLG), Universite Libre de Bruxelles, Bruxelles, Belgium
http://www.ulb.ac.be/di/mlg/

Maintainer: Benjamin Haibe-Kains
<bhaibeka@jimmy.harvard.edu>
<bhaibeka@ulb.ac.be>
Catharina Olsen
<colsen@ulb.ac.be>

adj.get.hops

Function to identify all children of a parent

Description
This function uses a depth-first search algorithm to identify all the children (and their corresponding depth) of a node.

Usage
adj.get.hops(adjmat)

Arguments
adjmat adjacency matrix; parents in rows, children in columns

Details
The algorithm is based on the depth-first search.

Value
two-column matrix containing the names of the children in the first column and their corresponding depth in the descent in the second column

Author(s)
Benjamin Haibe-Kains

Examples
## check whether a list of two nodes are children of another node
set.seed(54321)
mytopo <- matrix(sample(0:1, 100, replace=TRUE, prob=c(0.7,0.3)), nrow=10, dimnames=list(LETTERS[1:10], LETTERS[1:10]))
adj.get.hops(adjmat=mytopo)
adj.remove.cycles  

Function to remove cycles that may be present in a directed graph represented by an adjacency matrix

Description

This function removes cycles that may be present in a directed graph represented by an adjacency matrix,

Usage

adj.remove.cycles(adjmat, from, maxlength)

Arguments

adjmat  
adjacency matrix with positive entries represent evidence for the presence of an edge and entries less or equal than zero represent absence of an edge; parents in row, children in columns.

from  
indices or names of nodes for which the cycles present in the childhood should be removed; if missing, all cycles will be removed.

maxlength  
maximum length of path, once this length is reached no longer paths will be searched for.

Details

This function may be useful when it comes to generate a bayesian network using a topology identified from an source of information where cycles are allowed. When cycles are removed, the function tries to keep the most positive entries.

Value

A list of two items

adjmat.acyclic  
an adjacency matrix without cycles

adjmat.removed  
a matrix of booleans representing the edges that have been removed from the original adjacency matrix to make it acyclic

Author(s)

Benjamin Haibe-Kains

Examples

set.seed(54321)
xx <- matrix(sample(c(0,1), 100, replace=TRUE), nrow=10, ncol=10)
adj.remove.cycles(adjmat=xx, from=1, maxlength=3)
data.discretize

Function to discretize data based on user specified cutoffs

Description

This function enables discretization of data based on cutoffs specified by the users.

Usage

data.discretize(data, cuts)

Arguments

data: matrix of continuous or categorical values (gene expressions for example); observations in rows, features in columns.
cuts: list of cutoffs for each variable.

Details

This function is discretizing the continuous value in data using the cutoffs specified in cuts to create categories represented by increasing integers in 1,2,...,n where n is the maximum number of categories in the dataset.

Value

A matrix of categorical values where categories are {1,2,...,n} depending on the list of cutoffs specified in cuts; observations in rows, features in columns.

Author(s)

Benjamin Haibe-Kains

See Also

discretize

Examples

## load gene expression data for colon cancer data, list of genes related to RAS signaling pathway and the corresponding priors
data(expO.colon.ras)
## discretize the data in 3 categories
categories <- rep(3, ncol(data.ras))
## estimate the cutoffs (tertiles) for each gene
cuts.discr <- lapply(apply(rbind("nbcat"=categories, data.ras), 2, function(x) { y <- x[1]; x <- x[-1]; return(list(quantile(x=x, probs=seq(0, 1, length.out=y+1), na.rm=TRUE)[seq(1, y+1, 2)])) }), function(x) { return(x[[1]]) })
data.ras.bin <- data.discretize(data=data.ras, cuts=cuts.discr)
**eval.network**

*Function computing the f1-score, comparing an inferred topology with a given topology*

**Description**

This function computes the f1-score for an inferred topology using a topology provided by the user.

**Usage**

```r
eval.network(topo, true.topo)
```

**Arguments**

- `topo`  
  Inferred topology, an edge between variables X and Y corresponds to $\text{net}[X,Y]=1$.
- `true.topo`  
  Topology the user wants to compare the inferred topology with, e.g. the true network using generated datasets. An edge between variables X and Y corresponds to $\text{net}[X,Y]=1$.

**Value**

The computed f1-score, defined as $2 \times \text{TP} / (2 \times \text{TP} + \text{FN} + \text{FP})$.

**Author(s)**

Benjamin Haibe-Kains, Catharina Olsen

---

**expO.colon.ras**

*Gene expression, annotations, clinical data and priors for the colon cancer tumors collected by the expression project for oncology (expO).*

**Description**

This dataset contains (part of) the gene expression, annotations and clinical data as published by the expO project ([http://www.intgen.org/expo/](http://www.intgen.org/expo/)). Genes related to KRAS mutations were retrieved from Bild et al, Nature, 2006. Only genes with known gene symbols were selected resulting in a dataset of 292 human colon tumors and 259 RAS-related genes.

**Usage**

```r
data(expO.colon.ras)
```

**Format**

- `exp0.colon.ras` is a dataset containing four matrices:
  - `demo.ras`  
    Clinical information of the colon cancer patients whose tumors were hybridized.
  - `data.ras`  
    Matrix containing expression of genes related to RAS.
  - `annot.ras`  
    Matrix containing annotations of the genes related to RAS.
  - `priors.ras`  
    Matrix of priors counts for all the genes related to RAS. Each value represents the number of times an interaction was observed for a specific pair of genes (parents in rows, children in columns).
Details

The microarray platform used in the expO project is the Affymetrix HG-U133PLUS2 GeneChip.

Source

https://expo.intgen.org/geo/

References

http://www.intgen.org/expo/


Examples

data(expO.colon.ras)

Description

This dataset contains (part of) the gene expression, annotations and clinical data as published by Jorissen and colleagues in 2009. Genes related to KRAS mutations were retrieved from Bild et al, Nature, 2006. Only genes with known gene symbols were selected resulting in a dataset of 290 human colon tumors and 259 RAS-related genes.

Usage

data(jorissen.colon.ras)

Format

**jorissen.colon.ras** is a dataset containing four matrices:

- **demo2.ras** clinical information of the colon cancer patients whose tumors were hybridized.
- **data2.ras** matrix containing expression of genes related to RAS.
- **annot2.ras** matrix containing annotations of the genes related to RAS.
- **priors2.ras** matrix of priors counts for all the genes related to RAS. Each value represents the number of times an interaction was observed for a specific pair of genes (parents in rows, children in columns).

Details

The microarray platform used in Jorissen’s dataset is the Affymetrix HG-U133PLUS2 GeneChip.
Source


References


Examples

data(jorissen.colon.ras)

mcc

Function to compute the Matthews Correlation Coefficient (MCC) in a classification framework

Description

This function computes the Matthews Correlation Coefficient (MCC) in a classification framework.

Usage

mcc(ct, nbcat = nrow(ct))

Arguments

c t    contingency table
nbcat  number of categories

Value

MCC estimate

Author(s)

Benjamin Haibe-Kains
Function fitting a regression model for each gene in the data

Description

Function to fit a regression model for each variable in the dataset or alternatively each variable of interest.

Usage

net2pred(net, data, categories, predn, perturbations, method = c("linear", "linear.penalized", "cpt"), seed)

Arguments

net
  network object.

data
  matrix of continuous or categorical values (gene expressions for example); observations in rows, features in columns.

categories
  if this parameter missing, 'data' should be already discretized; otherwise either a single integer or a vector of integers specifying the number of categories used to discretize each variable (data are then discretized using equal-frequency bins) or a list of cutoffs to use to discretize each of the variables in 'data' matrix. If method='bayesnet' and categories is missing, data should contain categorical values and the number of categories will determine from the data.

predn
  indices or names of variables to fit during network inference. If missing, all the variables will be used for network inference.

perturbations
  matrix of 0,1 specifying whether a gene has been perturbed (e.g., knockdown, overexpression) in some experiments. Dimensions should be the same than data.

method
  type of predictive model to fit; linear for linear regression model, linear.penalized for regularized linear regression model, cpt for conditional probability tables estimated after discretization of the data.

seed
  set the seed to make the cross-validation and network inference deterministic.

Value

a new network object with the predictive models

Author(s)

Benjamin Haibe-Kains, Catharina Olsen

Examples

## load gene expression data for colon cancer data, list of genes related to RAS signaling pathway and the corresponding priors
data(expO.colon.ras)
## create matrix of perturbations (no perturbations in this dataset)
pert <- matrix(0, nrow=nrow(data.ras), ncol=ncol(data.ras), dimnames=dimnames(data.ras))

## number of genes to select for the analysis
genen <- 10
## select only the top genes
```r
go <- dimnames(annot.ras[[1]])[order(abs(log2(annot.ras[, "fold.change"])), decreasing=TRUE)[1:genen]]
mydata <- data.ras[ , go, drop=FALSE]
myannot <- annot.ras[goi, , drop=FALSE]
mydemo <- demo.ras
mypert <- pert[, goi, drop=FALSE]
```

## regression-based network inference

### infer global network from data and priors
```r
mynet <- netinf(data=mydata, perturbations=mypert, priors=mypriors, priors.count=TRUE, priors.weight=0.5, maxparents=3, method="regrnet", seed)
```

```r
net2pred(net=mynet, data=mydata, method="linear")
```

---

**netinf**

Function performing network inference by combining priors and genomic data

**Description**

Main function of the predictionet package, netinf infers a gene network by combining priors and genomic data. The two main network inference methodologies implemented so far are the bayesian and regression-based inferences.

**Usage**

```r
netinf(data, categories, perturbations, priors, predn, priors.count = TRUE, priors.weight = 0.5, maxparents = 3, method = "regrnet", seed)
```

**Arguments**

- **data**
  - matrix of continuous or categorical values (gene expressions for example); observations in rows, features in columns.

- **categories**
  - if this parameter missing, ‘data’ should be already discretized; otherwise either a single integer or a vector of integers specifying the number of categories used to discretize each variable (data are then discretized using equal-frequency bins) or a list of cutoffs to use to discretize each of the variables in ‘data’ matrix. If method=‘bayesnet’ and categories is missing, data should contain categorical values and the number of categories will determine from the data.

- **perturbations**
  - matrix of 0,1 specifying whether a gene has been perturbed (e.g., knockdown, overexpression) in some experiments. Dimensions should be the same than data.

- **priors**
  - matrix of prior information available for gene-gene interaction (parents in rows, children in columns). Values may be probabilities or any other weights (citations count for instance). If priors counts are used the parameter priors.count should be TRUE so the priors are scaled accordingly.

- **predn**
  - indices or names of variables to fit during network inference. If missing, all the variables will be used for network inference. Note that for bayesian network inference (method=‘bayesnet’) this parameter is ignored and a network will be generated using all the variables.
priors.count:
TRUE if priors specified by the user are number of citations (count) for each interaction. FALSE if probabilities or any other weight in [0,1] are reported instead.

priors.weight:
real value in [0,1] specifying the weight to put on the priors (0=only the data are used, 1=only the priors are used to infer the topology of the network).

maxparents:
maximum number of parents allowed for each gene.

subset:
vector of indices to select only subset of the observations.

method:
regrnet for regression-based network inference, bayesnet for bayesian network inference with the catnet package.

ensemble:
TRUE if the ensemble approach should be used, FALSE otherwise.

ensemble.model:
Could be either full or best depending how the equivalent networks are selected to be included in the ensemble network: for full bootstrapping is used to identify all the statistically equivalent networks, it best only the top ensemble.maxnsol are considered at each step of the feature selection.

ensemble.maxnsol:
maximum number of solutions to consider at each step of the feature selection for the method=ensemble.regrnet, default is 3.

causal:
'TRUE’ if the causality should be inferred from the data, 'FALSE’ otherwise

seed:
set the seed to make the network inference deterministic.

bayesnet.maxcomplexity:
maximum complexity for bayesian network inference, see Details.

bayesnet.maxiter:
maximum number of iterations for bayesian network inference, see Details.

verbose:
TRUE if messages should be printed, FALSE otherwise.

Details:
bayesnet.maxcomplexity and bayesnet.maxiter are parameters to be passed to the network inference method (see cnSearchOrder and cnSearchSA from the catnet package for more details). Relevance score is either MRMR scores if causal=FALSE or causality score if causal=FALSE.

Value:
method:
name of the method used for network inference.

ensemble:
is the network build using the ensemble approach?

topology:
adjaceny matrix representing the topology of the inferred network; parents in rows, children in columns.

topology.coeff:
if method=‘regrnet’ topology.coeff contains an adjacency matrix with the coefficients used in the local regression model; parents in rows, children in columns. Additionally the beta_0 values for each model in the first row of the matrix

dge.relevance:
relevance score for each edge (see Details).

Author(s):
Benjamin Haibe-Kains, Catharina Olsen
## load gene expression data for colon cancer data, list of genes related to RAS signaling pathway and the corresponding priors
```r
data(expO.colon.ras)
```

## create matrix of perturbations (no perturbations in this dataset)
```r
pert <- matrix(0, nrow=nrow(data.ras), ncol=ncol(data.ras), dimnames=dimnames(data.ras))
```

## number of genes to select for the analysis
```r
genen <- 10
```

## select only the top genes
```r
goi <- dimnames(annot.ras)[[1]][order(abs(log2(annot.ras[,"fold.change"])), decreasing=TRUE)[1:genen]]
```

## infer global network from data and priors
```r
mynet <- netinf(data=mydata, perturbations=mypert, priors=mypriors, priors.count=TRUE, priors.weight=0.5, maxparents=3, method="regrnet", seed=54321)
```

## plot network topology
```r
mytopo <- mynet$topology
library(network)
xnet <- network(x=mytopo, matrix.type="adjacency", directed=TRUE, loops=FALSE, vertex.attrnames=dimnames(mytopo)[[1]])
plot.network(x=xnet, displayisolates=TRUE, displaylabels=TRUE, boxed.labels=FALSE, label.pos=0, arrowhead.cex=2, vertex.cex=4, vertex.col="royalblue", jitter=FALSE, pad=0.5)
```

## export network as a 'gml' file that you can import into Cytoscape
```r
## Not run: rr <- netinf2gml(object=mynet, file="/predictionet_regrnet")
```

## infer a bayesian network network from data and priors
```r
## Not run: mynet <- netinf(data=mydata, perturbations=mypert, priors=mypriors, priors.count=TRUE, priors.weight=0.5, maxparents=3, method="bayesnet", seed=54321)
```

## plot network topology
```r
## Not run: mytopo <- mynet$topology
## Not run: library(network)
## Not run: xnet <- network(x=mytopo, matrix.type="adjacency", directed=TRUE, loops=FALSE, vertex.attrnames=dimnames(mytopo)[[1]])
## Not run: plot.network(x=xnet, displayisolates=TRUE, displaylabels=TRUE, boxed.labels=FALSE, label.pos=0, arrowhead.cex=2, vertex.cex=4, vertex.col="royalblue", jitter=FALSE, pad=0.5)
```

## export network as a 'gml' file that you can import into Cytoscape
```r
## Not run: rr <- netinf2gml(object=mynet, file="/predictionet_bayesnet")
```
Description

The function `netinf.cv` perform a cross-validation loop and infers a gene network by combining priors and genomic data in each fold. This allows to estimate the predictive ability of the inferred network as well as edge stability.

Usage

```r
netinf.cv(data, categories, perturbations, priors, predn, priors.count = TRUE, priors.weight = 0.5, ... "cpt"), nfold = 10, causal = TRUE, seed, bayesnet.maxcomplexity = 0, bayesnet.maxiter = 100, verbose = FALSE)
```

Arguments

data: matrix of continuous or categorical values (gene expressions for example); observations in rows, features in columns.

categories: if this parameter missing, 'data' should be already discretize; otherwise either a single integer or a vector of integers specifying the number of categories used to discretize each variable (data are then discretized using equal-frequency bins) or a list of cutoffs to use to discretize each of the variables in 'data' matrix. If method='bayesnet', this parameter should be specified by the user.

perturbations: matrix of 0, 1 specifying whether a gene has been perturbed (e.g., knockdown, overexpression) in some experiments. Dimensions should be the same than data.

priors: matrix of prior information available for gene-gene interaction (parents in rows, children in columns). Values may be probabilities or any other weights (citations count for instance). If priors counts are used the parameter priors.count should be TRUE so the priors are scaled accordingly.

predn: indices or names of variables to fit during network inference. If missing, all the variables will be used for network inference.

priors.count: TRUE if priors specified by the user are number of citations (count) for each interaction, FALSE if probabilities or any other weight in [0,1] are reported instead.

priors.weight: real value in [0,1] specifying the weight to put on the priors (0=only the data are used, 1=only the priors are used to infer the topology of the network).

maxparents: maximum number of parents allowed for each gene.

subset: vector of indices to select only subset of the observations.

method: regrnet for regression-based network inference, bayesnet for bayesian network inference with the catnet package.

ensemble: TRUE if the ensemble approach should be used, FALSE otherwise.

ensemble.maxnsol: Number of equivalent solutions chosen at each step.

predmodel: type of predictive model to fit; linear for linear regression model, linear.penalized for regularized linear regression model, cpt for conditional probability tables estimated after discretization of the data.

nfold: number of folds for the cross-validation.

causal: 'TRUE' if the causality should be inferred from the data, 'FALSE' otherwise.

seed: set the seed to make the cross-validation and network inference deterministic.

bayesnet.maxcomplexity: maximum complexity for bayesian network inference, see Details.

bayesnet.maxiter: maximum number of iterations for bayesian network inference, see Details.

verbose: TRUE if messages should be printed, FALSE otherwise.
Details

bayesnet.maxcomplexity and bayesnet.maxiter are parameters to be passed to the network inference method (see cnSearchOrder and cnSearchSA from the catnet package for more details).

Value

method name of the method used for network inference.

topology topology of the model inferred using the entire dataset.

topology.coeff if method='regrnet' topology.coeff contains an adjacency matrix with the coefficients used in the local regression model; parents in rows, children in columns. Additionally the beta_0 values for each model in the first row of the matrix.

topology.cv topology of the networks inferred at each fold of the cross-validation.

topology.coeff.cv if method='regrnet' topology.coeff contains an adjacency matrix with the coefficients used in the local regression model; parents in rows, children in columns. Additionally the beta_0 values for each model in the first row of the matrix. Inferred at each fold of the cross-validation.

topology.cv topology of the networks inferred at each fold of the cross-validation.

topology.coeff.cv if method='regrnet' topology.coeff contains an adjacency matrix with the coefficients used in the local regression model; parents in rows, children in columns. Additionally the beta_0 values for each model in the first row of the matrix. Inferred at each fold of the cross-validation.

topology.cv topology of the networks inferred at each fold of the cross-validation.

topology.coeff.cv if method='regrnet' topology.coeff contains an adjacency matrix with the coefficients used in the local regression model; parents in rows, children in columns. Additionally the beta_0 values for each model in the first row of the matrix. Inferred at each fold of the cross-validation.

prediction.score.cv list of prediction scores (R2, NRMSE, MCC) computed at each fold of the cross-validation.

topology.stability stability of the edges inferred during cross-validation; only the stability of the edges present in the network inferred using the entire dataset is reported.

topology.stability.cv stability of the edges inferred during cross-validation.

topology.relevance mean relevance score for each across folds in cross-validation.

topology.relevance.cv relevance score for each across computed during cross-validation.

Examples

## load gene expression data for colon cancer data, list of genes related to RAS signaling pathway and the corresponding priors
data(expO.colon.ras)
## create matrix of perturbations (no perturbations in this dataset)
pert <- matrix(0, nrow=nrow(data.ras), ncol=ncol(data.ras), dimnames=dimnames(data.ras))

## number of genes to select for the analysis
gen <- 10
## select only the top genes
goi <- dimnames(annot.ras)[[1]][order(abs(log2(annot.ras[, "fold.change"])), decreasing=TRUE)[1:gen]]
mydata <- data.ras[, goi, drop=FALSE]
myannot <- annot.ras[goi, , drop=FALSE]
mypriors <- priors.ras[goi, goi, drop=FALSE]
mydemo <- demo.ras
mypert <- pert[, goi, drop=FALSE]
# netinf.predict

Function to make prediction of a node values given its parents using an inferred network

## Description
This function predict the value of a node given its parents using an inferred network.

## Usage
```
netinf.predict(net, data, categories, perturbations, subset, predn, method=c("linear", "linear.penalized", "cpt"))
```

## Arguments
- **net**: a network object with local regression models.
- **data**: matrix of continuous or categorical values (gene expressions for example); observations in rows, features in columns.
- **categories**: if this parameter missing, 'data' should be already discretize; otherwise either a single integer or a vector of integers specifying the number of categories used to discretize each variable (data are then discretized using equal-frequency bins) or a list of cutoffs to use to discretize each of the variables in 'data' matrix. If method='bayesnet', this parameter should be specified by the user.
- **perturbations**: matrix of 0, 1 specifying whether a gene has been perturbed (e.g., knockdown, overexpression) in some experiments. Dimensions should be the same than data.
- **subset**: vector of indices to select only subset of the observations.
Function to create an igraph object and export a network to a GML readable by Cytoscape

Description

This function creates, from a network inferred from netinf or netinf.cv, an igraph object and export this network to a GML readable by Cytoscape.

Usage

netinf2gml(object, edge.info, node.info, file = "predictionet")
Arguments

object returns by netinf or netinf.cv

edge.info matrix of values representing the statistics for each edge; parents in rows, children in columns. A list of matrices could be provided, names of the list will then be used to describe the statistics in Cytoscape

node.info vector of values representing the statistics for each node; parents in rows, children in columns. A list of vectors could be provided, names of the list will then be used to describe the statistics in Cytoscape

file name of the GML file to be saved.

Details

The GML file created by this function has been tested on Cytoscape 2.8.1; a Vizmap property file of the same name is also created and could be imported into Cytoscape ("preditionet_vizmap2") so the information for each node and edge are displayed correctly.

Value

an igraph object

Author(s)

Benjamin Haibe-Kains

See Also

\codeRCytoscape

Examples

## load gene expression data for colon cancer data, list of genes related to RAS signaling pathway and the corresponding priors
data(expO.colon.ras)
## number of genes to select for the analysis
genen <- 10
## select only the top genes
goi <- dimnames(annot.ras)[[1]][order(abs(log2(annot.ras[,"fold.change"])), decreasing=TRUE)[1:genen]]
mydata <- data.ras[, goi, drop=FALSE]
myannot <- annot.ras[goi, , drop=FALSE]
mypriors <- priors.ras[goi, goi, drop=FALSE]
mydemo <- demo.ras
## infer global network from data and priors
mynet <- netinf.cv(data=mydata, categories=3, priors=mypriors, priors.count=TRUE, priors.weight=0.5, maxparents=3, method="regrnet", nfold=3, seed=54321)
## create an igraph object and export it into a GML file
## Not run: netinf2gml(object=mynet, file = "predictionet")
Function computing performance of prediction; methods include r2, nrmse and mcc

Description
This function computes prediction performance; methods include r2, nrmse and mcc.

Usage
pred.score(data, pred, categories, method = c("r2", "nrmse", "mcc"))

Arguments
data
pred
categories if this parameter missing, 'data' should be already discretize; otherwise either a single integer or a vector of integers specifying the number of categories used to discretize each variable (data are then discretized using equal-frequency bins) or a list of cutoffs to use to discretize each of the variables in 'data' matrix. If method='bayesnet', this parameter should be specified by the user.
method

Value
A vector of performance scores, one for each node

Author(s)
Benjamin Haibe-Kains, Catharina Olsen

See Also
netinf.predict

Examples
set.seed(54321)
xx <- runif(100)
## R2
pred.score(data=xx, pred=xx+rnorm(100)/10, method="r2")
## NRMSE
pred.score(data=xx, pred=xx+rnorm(100)/10, method="nrmse")
## MCC
pred.score(data=xx, pred=xx+rnorm(100)/10, categories=3, method="mcc")
predictionet.press.statistic

Function computing the press statistic for all target variables in topology

Description
The function predictionet.press.statistic computes the press statistic for all target variables in the provided topology.

Usage
predictionet.press.statistic(topo, data, ensemble=FALSE, perturbations=NULL)

Arguments
topo          adjacency matrix of 0,1 indicating whether two variables are connected
data          matrix of continuous or categorical values (gene expressions for example); observations in rows, features in columns.
perturbations matrix of 0, 1 specifying whether a gene has been perturbed (e.g., knockdown, overexpression) in some experiments. Dimensions should be the same than data.
ensemble      TRUE if the ensemble approach should be used, FALSE otherwise.

Value
A vector of press statistics, one for every target variable.

Author(s)
Benjamin Haibe-Kains, Catharina Olsen

Examples
```r
## load gene expression data for colon cancer data, list of genes related to RAS signaling pathway and the corresponding priors
data(expO.colon.ras)
## create matrix of perturbations (no perturbations in this dataset)
pert <- matrix(0, nrow=nrow(data.ras), ncol=ncol(data.ras), dimnames=dimnames(data.ras))

## number of genes to select for the analysis
genen <- 10
## select only the top genes
goi <- dimnames(annot.ras)[[1]][order(abs(log2(annot.ras[,"fold.change"])), decreasing=TRUE)[1:genen]]
mydata <- data.ras[, goi, drop=FALSE]
myannot <- annot.ras[goi, , drop=FALSE]
mpriors <- priors.ras[goi, goi, drop=FALSE]
mydemo <- demo.ras
mypert <- pert[, goi, drop=FALSE]

########################
## regression-based network inference
########################
```
## number of fold for cross-validation
res <- netinf.cv(data=mydata, categories=3, perturbations=mypert, priors=mypriors, priors.weight=0.5, method="regrnet", nfold=3, seed=54321)

## MCC for predictions in cross-validation
print(res$prediction.score.cv)

## export network as a 'gml' file that you can import into Cytoscape
## Not run: rr <- netinf2gml(object=res, file="predictionet_regrnet")

predictionet.stability.cv

### bayesian network inference

#### infer a bayesian network network from data and priors
## number of fold for cross-validation
## Not run: res <- netinf.cv(data=mydata, categories=3, perturbations=mypert, priors=mypriors, priors.count=TRUE, priors.weight=0.5, method="bayesnet", nfold=3, seed=54321)

#### MCC for predictions in cross-validation
## Not run: print(res$prediction.score.cv)

#### export network as a 'gml' file that you can import into Cytoscape
## Not run: rr <- netinf2gml(object=res, file="predictionet_bayesnet")

---

**predictionet.stability.cv**  
*Function inferring networks in cross-validation*

**Description**

The function *predictionet.stability.cv* infers networks in cross-validation (compared to net-inf.cv no regression is carried out, thus less computational cost but no prediction scores)

**Usage**

`predictionet.stability.cv(data, categories, perturbations, priors, predn, priors.count = TRUE, priors.weight = 0.5)`

**Arguments**

- **data**  
  matrix of continuous or categorical values (gene expressions for example); observations in rows, features in columns.

- **categories**  
  if this parameter missing, 'data' should be already discretize; otherwise either a single integer or a vector of integers specifying the number of categories used to discretize each variable (data are then discretized using equal-frequency bins) or a list of cutoffs to use to discretize each of the variables in 'data' matrix. If method='bayesnet', this parameter should be specified by the user.

- **perturbations**  
  matrix of 0, 1 specifying whether a gene has been perturbed (e.g., knockdown, overexpression) in some experiments. Dimensions should be the same than data.

- **priors**  
  matrix of prior information available for gene-gene interaction (parents in rows, children in columns). Values may be probabilities or any other weights (citations count for instance). if priors counts are used the parameter priors.count should be TRUE so the priors are scaled accordingly.
predictionet.stability.cv

predn: indices or names of variables to fit during network inference. If missing, all the variables will be used for network inference.

priors.count: TRUE if priors specified by the user are number of citations (count) for each interaction, FALSE if probabilities or any other weight in [0,1] are reported instead.

priors.weight: real value in [0,1] specifying the weight to put on the priors (0=only the data are used, 1=only the priors are used to infer the topology of the network).

maxparents: maximum number of parents allowed for each gene.

subset: vector of indices to select only subset of the observations.

method: regrnet for regression-based network inference, bayesnet for bayesian network inference with the catnet package.

ensemble: TRUE if the ensemble approach should be used, FALSE otherwise.

ensemble.maxnsol: Number of equivalent solutions chosen at each step.

nfold: number of folds for the cross-validation.

causal: 'TRUE' if the causality should be inferred from the data, 'FALSE' otherwise

seed: set the seed to make the cross-validation and network inference deterministic.

bayesnet.maxcomplexity: maximum complexity for bayesian network inference, see Details.

bayesnet.maxiter: maximum number of iterations for bayesian network inference, see Details.

Value

method: name of the method used for network inference.

topology: topology of the model inferred using the entire dataset.

topology.cv: topology of the networks inferred at each fold of the cross-validation.

edge.stability: stability of the edges inferred during cross-validation; only the stability of the edges present in the network inferred using the entire dataset is reported.

edge.stability.cv: stability of the edges inferred during cross-validation.

Author(s)

Benjamin Haibe-Kains, Catharina Olsen

Examples

```r
# load gene expression data for colon cancer data, list of genes related to RAS signaling pathway and the corresponding priors
data(expO.colon.ras)
# create matrix of perturbations (no perturbations in this dataset)
pert <- matrix(0, nrow=nrow(data.ras), ncol=ncol(data.ras), dimnames=dimnames(data.ras))

# number of genes to select for the analysis
genen <- 10

# select only the top genes
goi <- dimnames(annot.ras)[[1]][order(abs(log2(annot.ras[,"fold.change"])), decreasing=TRUE)[1:genen]]
mydata <- data.ras[, goi, drop=FALSE]
myannot <- annot.ras[goi, , drop=FALSE]
mypriors <- priors.ras[goi, goi, drop=FALSE]
```
mydemo <- demo.ras
mypert <- pert[, goi, drop=FALSE]

# regression-based network inference
# number of fold for cross-validation
res <- netinf.cv(data=mydata, categories=3, perturbations=mypert, priors=mypriors, priors.weight=0.5, method="regrnet", nfold=3, seed=54321)

# MCC for predictions in cross-validation
print(res$prediction.score.cv)

# export network as a 'gml' file that you can import into Cytoscape
# Not run: rr <- netinf2gml(object=res, file="predictionet_regrnet")

# bayesian network inference
# infer a bayesian network network from data and priors
# number of fold for cross-validation
# Not run: res <- netinf.cv(data=mydata, categories=3, perturbations=mypert, priors=mypriors, priors.count=TRUE, priors.weight=0.5, method="bayesnet", nfold=3, seed=54321)

# MCC for predictions in cross-validation
# Not run: print(res$prediction.score.cv)

# export network as a 'gml' file that you can import into Cytoscape
# Not run: rr <- netinf2gml(object=res, file="predictionet_bayesnet")
Index

*Topic **classif**
  mcc, 8
  netinf.predict, 15
  pred.score, 18
*Topic **data**
  expO.colon.ras, 6
  jorissen.colon.ras, 7
*Topic **graphs**
  adj.remove.cycles, 4
  eval.network, 6
  net2pred, 9
  netinf, 10
  netinf.cv, 12
  predictionet.press.statistic, 19
  predictionet.stability.cv, 20
*Topic **graph**
  netinf2gml, 16
*Topic **package**
  predictionet-package, 2
*Topic **regression**
  netinf.predict, 15
  pred.score, 18
*Topic **univar**
  mcc, 8
  pred.score, 18

adj.get.hops, 3
adj.remove.cycles, 4
annot.ras(expO.colon.ras), 6
annot2.ras(jorissen.colon.ras), 7
cnSearchOrder, 11, 14
cnSearchSA, 11, 14
data.discretize, 5
data.ras(expO.colon.ras), 6
data2.ras(jorissen.colon.ras), 7
demo.ras(expO.colon.ras), 6
demo2.ras(jorissen.colon.ras), 7
discretize, 5
eval.network, 6
expO.colon.ras, 6, 6
igraph, 16, 17

jorissen.colon.ras, 7, 7
mcc, 8
net2pred, 9
netinf, 10, 16
netinf.cv, 12, 16
netinf.predict, 15, 18
netinf2gml, 16
pred.score, 18
predictionet(predictionet-package), 2
predictionet-package, 2
predictionet.press.statistic, 19
predictionet.stability.cv, 20
priors.ras(expO.colon.ras), 6
priors2.ras(jorissen.colon.ras), 7