**Aracne**

*Algorithm for the Reconstruction of Accurate Cellular Networks*

---

**Description**

This function takes the mutual information matrix as input in order to return the inferred network according to the Aracne algorithm. This algorithm applies the data processing inequality to all triplets of nodes in order to remove the least significant edge in each triplet.

**Usage**

```r
aracne( mim, eps=0 )
```

**Arguments**

- `mim`: A square matrix whose i,j th element is the mutual information between variables $X_i$ and $X_j$ - see `build.mim`
- `eps`: Numeric value indicating the threshold used when removing an edge: for each triplet of nodes (i,j,k), the weakest edge, say (ij), is removed if its weight is below min{(ik),(jk)}-eps - see references.

**Details**

The Aracne procedure starts by assigning to each pair of nodes a weight equal to their mutual information. Then, the weakest edge of each triplet is interpreted as an indirect interaction and is removed if the difference between the two lowest weights is above a threshold `eps`.

**Value**

`aracne` returns a matrix which is the weighted adjacency matrix of the network. In order to display the network, load the package `Rgraphviz` and use the following command:

```r
plot( as( returned.matrix ,"graphNEL") )
```

**References**

**build.mim**

**See Also**

`build.mim, clr, mrnet`

**Examples**

```r
data(syn.data)
mim <- build.mim(discretize(syn.data))
net <- aracne(mim)
```

**build.mim**  
*Build Mutual Information Matrix*

**Description**

`build.mim` takes the dataset as input and computes the mutual information between all pairs of variables according to the mutual information estimator `estimator`. The results are saved in the mutual information matrix (MIM), a square matrix whose \((i,j)\) element is the mutual information between variables \(X_i\) and \(X_j\).

**Usage**

```r
build.mim(data, estimator="mi.empirical")
```

**Arguments**

- `data`: data.frame containing gene expression data or any dataset where columns contain variables/features and rows contain outcomes/samples.
- `estimator`: The name of the mutual information estimator. The package implements four estimators: "mi.empirical", "mi.mm", "mi.shrink", "mi.sg" (default: "mi.empirical") - see details. These estimators require discrete data values - see `discretize`.

**Details**

- "mi.empirical": This estimator computes the entropy of the empirical probability distribution.
- "mi.mm": This is the Miller-Madow asymptotic bias corrected empirical estimator.
- "mi.shrink": This is a shrinkage estimate of the entropy of a Dirichlet probability distribution.
- "mi.sg": This is the Schurmann-Grassberger estimate of the entropy of a Dirichlet probability distribution.

**Value**

`build.mim` returns the mutual information matrix.

**Author(s)**

Patrick E. Meyer, Frederic Lafitte, Gianluca Bontempi
References


See Also
clr, aracne, mrnet

Examples

data(syn.data)
#mutual information estimator
estimator="mi.empirical"
#number of bins used to discretize
nb.bins = sqrt(nrow(syn.data))
mim <- build.mim(discretize(syn.data,nbins=nb.bins),estimator)

clr  Context Likelihood or Relatedness Network

Description

clr takes the mutual information matrix as input in order to return the inferred network - see details.

Usage

clr( mim )

Arguments

mim  A square matrix whose i,j th element is the mutual information between variables Xi and Xj - see build.mim.

Details

The CLR algorithm is an extension of relevance network. Instead of considering the mutual information \( I(X_i; X_j) \) between features \( X_i \) and \( X_j \), it takes into account the score \( \sqrt{z_i^2 + z_j^2} \), where

\[
z_i = \max \left\{ 0, \frac{I(X_i; X_j) - \mu_i}{\sigma_i} \right\}
\]

and \( \mu_i \) and \( \sigma_i \) are, respectively, the mean and the standard deviation of the empirical distribution of the mutual information values \( I(X_i; X_k), k=1,...,n \).
discretize

discretize returns a matrix which is the weighted adjacency matrix of the network. In order to display the network, load the package Rgraphviz and use the following command: plot(as(returned.matrix, "graphNEL"))

References

See Also
build.mim, aracne, mrnet

Examples
```r
data(syn.data)
mim <- build.mim(discretize(syn.data))
net <- clr(mim)
```

Description
discretize discretizes data using the equal frequencies or equal width binning algorithm.

Usage
```r
discretize(data, disc = "equalfreq", nbins = sqrt(nrow(data)))
```

Arguments
- `data` A data.frame containing data to be discretized. The columns contains variables and the rows samples.
- `disc` The name of the discretization method to be used: "equalfreq" or "equalwidth" (default: "equalfreq") - see references.
- `nbins` Integer specifying the number of bins to be used for the discretization. By default the number of bins is set to $\sqrt{N}$ where N is the number of samples.

Value
discretize returns the discretized dataset.

Author(s)
Patrick E. Meyer, Frederic Lafitte, Gianluca Bontempi
**minet**  

**Mutual Information Network**

**Description**

For a given dataset, **minet** infers the network in two steps. First, the mutual information between all pairs of variables in *dataset* is computed according to the *estimator* argument. Then the algorithm given by *method* considers the estimated mutual informations in order to build the network.

**Usage**

```r
minet(dataset, method="mrnet", estimator="mi.empirical",
      disc="equalfreq", nbins=sqrt(nrow(dataset)) )
```

**Arguments**

- `dataset` data.frame where columns contain variables/features and rows contain outcomes/samples.
- `method` The name of the inference algorithm : "clr", "aracne" or "mrnet" (default: "mrnet") - see references.
- `estimator` The name of the mutual information estimator : "mi.empirical", "mi.mm", "mi.shrink" or "mi.sg"(default: "mi.empirical") - see **build.mim**.
- `disc` The name of the discretization method to be used : "equalfreq" or "equalwidth" (default: "equalfreq") - see **discretize**.
- `nbins` Integer giving the number of bins to be used in the discretization.

**Value**

**minet** returns a matrix which is the weighted adjacency matrix of the network. The weights range from 0 to 1 and can be seen as a confidence measure on the presence of the arcs. In order to display the network, load the package Rgraphviz and use the following command:

```r
plot( as( returned.matrix ,"graphNEL") )
```

**Author(s)**

Patrick E. Meyer, Frederic Lafitte, Gianluca Bontempi

---

**References**


**See Also**

**build.mim**

**Examples**

```r
data(syn.data)
ew.data <- discretize(syn.data,"equalwidth")
ef.data <- discretize(syn.data,"equalfreq")
```
References


See Also

build.mim, clr, mrnet, aracne

Examples

data(syn.data)
net1 <- minet( syn.data )
net2 <- minet( syn.data, estimator="mi.shrink" )
net3 <- minet( syn.data, method="clr", estimator="mi.sg" )

mrnet

Maximum Relevance Minimum Redundancy

Description

mrnet takes the mutual information matrix as input in order to infer the network using the maximum relevance/minimum redundancy feature selection method - see details.

Usage

mrnet( mim )

Arguments

mim A square matrix whose i,j th element is the mutual information between variables $X_i$ and $X_j$ - see build.mim.

Details

Consider a supervised learning task, where the output is denoted by $Y$ and $V$ is the set of input variables. The method ranks the set $V$ of inputs according to a score that is the difference between the mutual information with the output variable $Y$ (maximum relevance) and the average mutual information with the previously ranked variables (minimum redundancy). The greedy search starts by selecting the variable $X_i$ having the highest mutual information with the target $Y$. The second selected variable $X_j$ will be the one that maximizes $I(X_j; Y) - I(X_j; X_i)$. In the following steps, given a set $S$ of selected variables, the criterion updates $S$ by choosing the variable $X_k$ that maximizes $I(X_k; Y) - \frac{1}{|S|} \sum_{i \in S} I(X_k; X_i)$.

The MRNET approach consists in repeating this selection procedure for each target variable by
putting \( Y = X_i \) and \( V = X \setminus \{X_i\}, i=1,...,n \) where \( X \) is the set of outcomes of all variables. The weight of each pair \( X_i, X_j \) will be the maximum score between the one computed when \( X_i \) is the output and the one computed when \( X_j \) is the output.

**Value**

`mrnet` returns a matrix which is the weighted adjacency matrix of the network. In order to display the network, load the package `Rgraphviz` and use the following command:

```
plot( as( returned.matrix ,"graphNEL") )
```

**Author(s)**

Patrick E. Meyer, Frederic Lafitte, Gianluca Bontempi

**References**


**See Also**

`build.mim`, `clr`, `aracne`

**Examples**

```r
data(syn.data)
mim <- build.mim(discretize(syn.data))
net <- mrnet(mim)
```

---

**norm**

**Data Normalization**

**Description**

Normalizes data \( x \) using the following code: \( (x - \min(x)) / (\max(x) - \min(x)) \).

**Usage**

`norm(x)`

**Arguments**

\( x \)  
A data.frame or matrix or vector containing the data to be normalized. In this package the `norm` method is used to normalize the network’s weighted adjacency matrices.

**Value**

`norm` returns the data.frame \( x \) with normalized values (i.e. values ranging from 0 to 1)

**Examples**

```r
data <- runif(100,-10,10)
ndata <- norm(data)
```
**syn.data  
Simulated Gene Expression Data**

**Description**

Dataset containing 100 samples and 50 genes generated by the publicly available SynTReN generator using a yeast source network - see `syn.net`.

**Usage**

```r
data(syn.data)
```

**Format**

`syn.data` is a data frame containing 100 rows and 50 columns. Each row contains a microarray experiment and each column contains a gene.

**Source**

SynTReN 1.1.3

source network : yeast_nn.sif

**References**


**Examples**

```r
data(syn.data)
data(syn.net)
syn.data <- discretize(syn.data)
mim <- build.mim(syn.data)
inferred.net <- mrnet(mim)
max(fscores(validate(inferred.net, syn.net )))
```

---

**syn.net  
SynTReN Source Network**

**Description**

This is the true underlying network used to generate the dataset loaded by `data(syn.data)` - see `syn.data`.

**Usage**

```r
data(syn.net)
```
**validate**

**Format**

`syn.net` is a boolean adjacency matrix representing an undirected graph of 50 nodes.

**Source**

`syn.net` is the "yeast_nn.sif" source network from the SynTReN generator where all the variables/nodes not in `syn.data` were removed.

**References**


**Examples**

```r
data(syn.data)
data(syn.net)
mim <- build.mim(discretize(syn.data), estimator="mi.empirical")
inferred.net <- mrnet(mim)
max(fscores(validate(inferred.net, syn.net )))
```

---

**validate**  
*Inference Validation*

**Description**

`validate` compares the inferred network to the true underlying network for several threshold values and appends the resulting confusion matrices to the returned object.

**Usage**

```r
validate( inet, tnet, steps=50 )
```

**Arguments**

- `inet`  
  This is the inferred network, a data.frame or matrix obtained by one of the functions `minet`, `aracne`, `clr` or `mrnet`.

- `tnet`  
  The true underlying network. This network must have the same size and variable names as `inet`.

- `steps`  
  The number of threshold values to be used in the validation process - see `details`.

**Details**

For each of the `steps` threshold values $I_0$, the edges whose weight are (strictly) below $I_0$ are eliminated. All the other edges will have a weight 1. Thus for each threshold, we obtain a boolean network from the inferred network. This network is compared to the true underlying network, `tnet`, in order to compute a confusion (adjacency) matrix. All the confusion matrices, obtained with different threshold values, are appended to the returned object. In the end the `validate` function returns a data.frame containing `steps` confusion matrices.
Value

`validates returns a data.frame with four columns named thrsh, tp, fp, fn. These values are computed for each of the steps thresholds. Thus each row of the returned object contains the confusion matrix for a different threshold.

See Also

`minet,vis.res`

Examples

data(syn.data)
data(syn.net)
inf.net <- mrnet(build.mim(discretize(syn.data)))
table <- validate( inf.net, syn.net, steps=100 )

vis.res Visualize Results

Description

A group of functions to plot precision-recall and ROC curves and to compute f-scores from the data.frame returned by the `validate` function.

Usage

`pr(table)`
`rates(table)`
`fscores(table, beta=1)`
`show.pr(table, device=-1,...)`
`show.roc(table, device=-1,...)`

Arguments

table This is the (steps x 5) data.frame returned by the `validate` function where steps is the number of thresholds used in the validation process and where columns contain TP,FP,TN,FN values (confusion matrix) as well as the threshold value used - see `validate`.

beta Numeric used as the weight of the recall in the f-score formula - see details. The default value of this argument is 1, meaning precision as important as recall.

device The device to be used. This parameter allows the user to plot precision-recall and receiver operating characteristic curves for various inference algorithms on the same plotting window - see examples.

... arguments passed to `plot`
Details

A confusion matrix contains FP, TP, FN, FP values.

"true positive rate" \( tpr = \frac{TP}{TN+TP} \)

"false positive rate" \( fpr = \frac{FP}{FN+FP} \)

"precision" \( p = \frac{TP}{TP+FP} \)

"recall" \( r = \frac{TP}{TP+FN} \)

"f-beta-score" \( F_\beta = (1 + \beta^2) \frac{pr \cdot r}{\beta^2 pr + r} \)

Value

The function `show.roc` (or `show.pr`) plots the ROC-curve (PR-curve) and returns the device associated with the plotting window.

The function `pr` returns a \((\text{steps} \times 2)\) data.frame where \text{steps} is the number of thresholds used in the validation process. The first column contains precisions and the second recalls - see details.

The function `rates` also returns a \((\text{steps} \times 2)\) data.frame where the first column contains true positive rates and the second column false positive rates - see details.

The function `fscores` returns \text{steps} fscores according to the \text{steps} confusion matrices contained in the ‘table’ argument - see details.

References


See Also

`validate`, `plot`

Examples

data(syn.data)
data(syn.net)

# Inference
mr <- minet( syn.data, method="mrnet", estimator="mi.empirical" )
ar <- minet( syn.data, method="aracne", estimator="mi.empirical" )
clr<- minet( syn.data, method="clr", estimator="mi.empirical" )

# Validation
mr.tbl <- validate(mr,syn.net)
ar.tbl <- validate(ar,syn.net)
clr.tbl<- validate(clr,syn.net)

# Plot PR-Curves
max(fscores(mr.tbl))
dev <- show.pr(mr.tbl, col="green", type="b")
dev <- show.pr(ar.tbl, device=dev, col="blue", type="b")
show.pr(clr.tbl, device=dev, col="red",type="b")
Index

*Topic datasets
  syn.data, 8
  syn.net, 8
*Topic misc
  aracne, 1
  build.mim, 2
  clr, 3
  discretize, 4
  minet, 5
  mrnet, 6
  norm, 7
  validate, 9
  vis.res, 10
aracne, 1, 3, 4, 6, 7, 9
build.mim, 1, 2, 2–7
clr, 2, 3, 3, 6, 7, 9
discretize, 2, 4, 5
fscores(vis.res), 10
minet, 5, 9, 10
mrnet, 2–4, 6, 6, 9
norm, 7
plot, 11
pr(vis.res), 10
rates(vis.res), 10
show.pr(vis.res), 10
show.roc(vis.res), 10
syn.data, 8, 8
syn.net, 8, 8
validate, 9, 10, 11
vis.res, 10, 11