Package ‘pathClass’

Type Package

Title Classification using biological pathways as prior knowledge

Version 0.9.4

Date 2013-06-25

Author Marc Johannes

Maintainer Marc Johannes <JohannesMarc@gmail.com>

Description pathClass is a collection of classification methods that use information about feature connectivity in a biological network as an additional source of information. This additional knowledge is incorporated into the classification a priori. Several authors have shown that this approach significantly increases the classification performance.

Depends R (>= 2.14), svmpath, kernlab, affy, Biobase, ROCR, igraph, lpSolve

Enhances parallel

Suggests hu6800.db, golubEsets

License GPL (>= 2)

LazyLoad yes

Collate 'CrossValidation.R' 'GeneRank.R' 'GraphSVM.R'
  'networkBasedSVM.R' 'PathwayMethods.R'
  'RecursiveFeatureElimination.R' 'RRFE.R' 'SpanBound.R' 'SVMs.R'

Repository CRAN

Date/Publication 2013-07-01 07:43:40

NeedsCompilation no

R topics documented:

pathClass-package .................................................. 2
adjacency.matrix .................................................. 3
as.adjacencyList .................................................. 3
calc.diffusionKernel ............................................. 4
Classification with SMVs and prior knowledge

Details

pathClass is a collection of classification methods that use information about how features are connected in the underlying biological network as an additional source of information. This additional knowledge is incorporated into the classification a priori. Several authors have shown that this approach significantly increases the classification performance.

Author(s)

Marc Johannes <JohannesMarc@gmail.com>
adjacency.matrix  An adjacency matrix of a random graph

Description

An adjacency matrix of a random graph with some random Refseq Protein IDs for use in example files and the vignette

Author(s)

Marc Johannes <JohannesMarc@gmail.com>

as.adjacencyList  Uses a adjacency matrix to create a adjacency list

Description

Uses a adjacency matrix to create a adjacency list as needed for fit.networkBasedSVM.

Usage

as.adjacencyList(adjacency.matrix, skip.redundant.nodes = TRUE, is.directed = FALSE)

Arguments

adjacency.matrix
a adjacency matrix.

skip.redundant.nodes
if TRUE and the graph is undirected only the upper triangular matrix (including the diagonal) is used to create the adjacency list.

is.directed
determines wether or not the graph is directed.

Value

an adjacency list.

Author(s)

Marc Johannes <JohannesMarc@gmail.com>
Examples

```r
## Not run:
library(pathClass)
data(adjacency.matrix)
ad.list <- as.adjacencyList(adjacency.matrix)

## End(Not run)
```

calc.diffusionKernel  Calculation of diffusion kernel matrix

Description

Calculation of diffusion kernel matrix

Usage

```r
calc.diffusionKernel(L, is.adjacency = FALSE, beta = 0)
```

Arguments

- `L`: Laplacian or transition probability matrix
- `is.adjacency`: is `L` a laplace or adjacency matrix
- `beta`: beta parameter of the diffusion kernel. beta controls the extent of diffusion.

Value

the diffusion kernel

Author(s)

Marc Johannes <JohannesMarc@gmail.com>

References

crossval

Performs cross-validation with a specified algorithm

Description

Performs a cross-validation using the specified algorithms. If package parallel is loaded the cross-validation will be performed in parallel. If the parallel package is loaded but a parallel cross-validation is not wanted parallel can be set to FALSE. If parallel cross-validation is desired the number of cores can be choosen by using the cores parameter.

Usage

crossval(x, y, theta.fit, folds = 10, repeats = 1, parallel = TRUE, cores = NULL, DEBUG = FALSE, ...)

Arguments

x  
a p x n matrix of expression measurements with p samples and n genes.
y  
a factor of length p comprising the class labels.
theta.fit  
the method to learn a decision boundary. Currently available are fit.rrfe, fit.rfe, fit.graph.svm, fit.networkBasedSVM
folds  
number of folds to perform
repeats  
number of how often to repeat the x-fold cross-validation
parallel  
should the cross-validation be performed in parallel i.e. on several cpu-cores. (see also Details section)
cores  
specify the number of cores that should be used for parallel cross-validation.
DEBUG  
should debugging information be plotted. Defaults to n - 1 cores.
...
additional parameters to theta fit.

Value

a list with the results of the cross-validation. See details for more information.

Note

Parallel cross-validation can only be performed if the parallel-package was loaded prior to calling this function.

Author(s)

Marc Johannes <JohannesMarc@gmail.com>

See Also

fit.rrfe, fit.rfe, fit.graph.svm, fit.networkBasedSVM
Examples

```r
## Not run:
set.seed(4321)
data(example_data)
res.rfe <- crossval(x, y, DEBUG=TRUE, theta.fit=fit.rfe, folds=2, repeats=1, parallel=TRUE,
                    Cs=10^(-3:3))
res.rrfe <- crossval(x, y, DEBUG=TRUE, theta.fit=fit.rre, folds=3, repeats=1, parallel=TRUE,
                    Cs=10^(-3:3), mapping=mapping, Gsub=adjacency.matrix, d=1/2)

## End(Not run)
```

desummarize.ranks  
Desummarize GeneRanks back to the corresponding probesets

Description

Desummarize the GeneRanks which were previously calculated for each node in the underlying biological network back to the corresponding probesets for the Reweighted Recursive Feature Elimination (RRFE).

Usage

```r
desummarize.ranks(ranks, mapping)
```

Arguments

- `ranks`  
  the previously calculated GeneRanks or PageRanks.
- `mapping`  
  a matrix or data.frame with 2 columns. The colnames of mapping have to contain at least 'graphID' and 'probesetID'.

Value

- matrix with 1st column probeIDs 2nd column gene IDs

extractFeatures  
Extracts features which have been choosen by the classifier(s).

Description

This function extracts the features which have been selected by the classifiers during the cross-validation along with the number of times they have been choosen. When, for example, performing a 5 times repeated 10-fold cross-validation the maximum number a feature can be choosen is 50.

Usage

```r
extractFeatures(res, toFile = FALSE,
               fName = "ClassificationFeatures.csv")
```
**Arguments**

- **res** A result of `crossval`.
- **toFile** Should the results be printed into a CSV-file.
- **fName** the name of the file to save the results in.

**Value**

a data.frame indicating the number of times a feature has been chosen.

**Author(s)**

Marc Johannes <JohannesMarc@gmail.com>

**Examples**

```r
## Not run:
library(Biobase)
data(sample.ExpressionSet)
x <- t(exprs(sample.ExpressionSet))
y <- factor(pData(sample.ExpressionSet)$sex)
res.rfe <- crossval(x,y,DEBUG=TRUE,theta.fit=fit.rfe,folds=2, repeats=1, parallel=TRUE,Cs=10^c(-3:3))
eextractFeatures(res.rfe, toFile=FALSE)

## End(Not run)
```

---

**Description**

mapping must be a data.frame with at least two columns. The column names have to be c('probesetID', 'graphID'). Where 'probesetID' is the probeset ID present in the expression matrix (i.e. `colnames(x)`) and 'graphID' is any ID that represents the nodes in the diffusionKernel (i.e. `colnames(diffusionKernel)` or `rownames(diffusionKernel)`). The purpose of this mapping is that a gene or protein in the network might be represented by more than one probe set on the chip. Therefore, the algorithm must know which genes/protein in the network belongs to which probeset on the chip.

**Usage**

```r
fit.graph.svm(x, y, DEBUG = FALSE,
scale = c("center", "scale"), Cs = 10^c(-3:3),
stepsize = 0.1, mapping, diffusionKernel,
useOrigMethod = FALSE)
```
Arguments

- **x**: A p x n matrix of expression measurements with p samples and n genes.
- **y**: A factor of length p comprising the class labels.
- **DEBUG**: Should debugging information be plotted.
- **scale**: A character vector defining if the data should be centered and/or scaled. Possible values are center and/or scale. Defaults to c('center', 'scale').
- **Cs**: Soft-margin tuning parameter of the SVM. Defaults to 10*c(-3:3).
- **stepsize**: Amount of features that are discarded in each step of the feature elimination. Defaults to 10%.
- **mapping**: A mapping that defines how probe sets are summarized to genes.
- **diffusionKernel**: The diffusion kernel which was pre-computed by using the function `calc.diffusionKernel`.
- **useOrigMethod**: Use the method originally described in the paper by Franck Rapaport et al. 2007.

Value

- A `graphSVM` object
  - **features**: The selected features
  - **errorNbound**: The span bound of the model
  - **fit**: The fitted SVM model

Note

We combined the original method with a Recursive Feature Elimination in order to allow a feature selection. The optimal number of features is found by using the span estimate. See Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1), 131-159.

Author(s)

Marc Johannes <JohannesMarc@gmail.com>

References


Examples

```r
## Not run:
library(Biobase)
data(sample.ExpressionSet)
x <- t(exprs(sample.ExpressionSet))
y <- factor(pData(sample.ExpressionSet)$sex)
# create the mapping
library('hgu95av2.db')
```
fit.networkBasedSVM


Description

mapping must be a data.frame with at least two columns. The column names have to be c('probesetID', 'graphID'). Where 'probesetID' is the probeset ID present in the expression matrix (i.e. colnames(x)) and 'graphID' is any ID that represents the nodes in the diffusionKernel (i.e. colnames(diffusionKernel) or rownames(diffusionKernel)). The purpose of this mapping is that a gene or protein in the network might be represented by more than one probe set on the chip. Therefore, the algorithm must know which genes/protein in the network belongs to which probeset on the chip.

Usage

```r
fit.networkBasedSVM(exps, y, DEBUG = FALSE, n.inner = 3,
scale = c("center", "scale"), sd.cutoff = 1,
lambda = 10^(-2:4), adjacencyList)
```

Arguments

- `exps`: a p x n matrix of expression measurements with p samples and n genes.
- `y`: a factor of length p comprising the class labels.
- `DEBUG`: should debugging information be plotted.
- `n.inner`: number of fold for the inner cross-validation.
- `scale`: a character vector defining if the data should be centered and/or scaled. Possible values are `center` and/or `scale`. Defaults to `c('center', 'scale')`.
- `sd.cutoff`: a cutoff on the standard deviation (sd) of genes. Only genes with sd > sd.cutoff stay in the analysis.
- `lambda`: a set of values for lambda regularization parameter of the L∞-Norm. Which, if properly chosen, eliminates factors that are completely irrelevant to the response, what in turn leads to a factor-wise (subnetwork-wise) feature selection. The 'best' lambda is found by an inner-cross validation.
adjacencyList  a adjacency list representing the network structure. The list can be generated from a adjacency matrix by using the function as.adjacencyList

Value

a networkBasedSVM object containing

features the selected features

lambda.performance

overview how different values of lambda performed in the inner cross validation

fit

the fitted network based SVM model

Author(s)

Marc Johannes <JohannesMarc@gmail.com>

References


Examples

```r
## Not run:
library(Biobase)
data(sample.ExpressionSet)
x <- t(exprs(sample.ExpressionSet))
y <- factor(pData(sample.ExpressionSet)$sex)

# create the mapping
library('hgu95av2.db')
mapped.probes <- mappedkeys(hgu95av2REFSEQ)
refseq <- as.list(hgu95av2REFSEQ[[mapped.probes]])
times <- sapply(refseq, length)
mapping <- data.frame(probesetID=rep(names(refseq), times), times=times, graphID=unlist(refseq), row.names=NULL, stringsAsFactors=FALSE)
mapping <- unique(mapping)
library(pathClass)
data(adjacency.matrix)
matched <- matchMatrices(x=x, adjacency=adjacency.matrix, mapping=mapping)
ad.list <- as.adjacencyList(matched$adjacency)
res.nBSVM <- crossval(matched$x, y, theta.fit=fit.networkBasedSVM, folds=3, repeats=1, DEBUG=TRUE, parallel=FALSE, adjacencyList=ad.list, lambdas=10^(-1:2), sd.cutoff=50)

## End(Not run)
```
Recursive Feature Elimination (RFE)

Description

Implementation of the Recursive Feature Elimination (RFE) algorithm.

Usage

```r
fit.rfe(x, y, DEBUG = FALSE,
    scale = c("center", "scale"), Cs = 10^c(-3:3),
    stepsize = 0.1)
```

Arguments

- `x` a p x n matrix of expression measurements with p samples and n genes.
- `y` a factor of length p comprising the class labels.
- `DEBUG` should debugging information be plotted.
- `scale` a character vector defining if the data should be centered and/or scaled. Possible values are `center` and/or `scale`. Defaults to `c('center', 'scale')`.
- `Cs` soft-margin tuning parameter of the SVM. Defaults to `10^c(-3:3)`.
- `stepsize` amount of features that are discarded in each step of the feature elimination. Defaults to 10%.

Value

a RFE fit object. `features` = selected features `error.bound` = span bound of the model `fit` = fitted SVM model

Note


Author(s)

Marc Johannes <JohannesMarc@gmail.com>

Examples

```r
# Not run:
library(BioBase)
data(sample.ExpressionSet)
x <- t(exprs(sample.ExpressionSet))
y <- factor(pData(sample.ExpressionSet)$sex)
res.rfe <- crossval(x, y,DEBUG=TRUE,theta.fit=fit.rfe,folds=2,repeats=1,parallel=TRUE,Cs=10^(-3:3))
```
Reweighted Recursive Feature Elimination (RRFE)

Description

Implementation of the Reweighted Recursive Feature Elimination (RRFE) algorithm. `mapping` must be a data.frame with at least two columns. The column names have to be `c('probesetID', 'graphID')`. Where ‘probesetID’ is the probeset ID present in the expression matrix (i.e. `colnames(x)`) and 'graphID' is any ID that represents the nodes in the graph (i.e. `colnames(Gsub)` or `rownames(Gsub)`). The purpose of the this mapping is that a gene or protein in the network might be represented by more than one probe set on the chip. Therefore, the algorithm must know which genes/protein in the network belongs to which probe set on the chip. However, the method is able to use all feature when one sets the parameter `useAllFeatures` to TRUE. When doing so, RRFE assigns the minimal weight returned by GeneRank to those genes which are not present in Gsub.

Usage

```r
fit.rrfe(x, y, DEBUG = FALSE,
    scale = c('center', 'scale'), Cs = 10^c(-3:3),
    stepsize = 0.1, useAllFeatures = F, mapping, Gsub,
    d = 0.5)
```

Arguments

- `x`: a p x n matrix of expression measurements with p samples and n genes.
- `y`: a factor of length p comprising the class labels.
- `DEBUG`: should debugging information be plotted.
- `scale`: a character vector defining if the data should be centered and/or scaled. Possible values are `center` and/or `scale`. Defaults to `c('center', 'scale')`.
- `Cs`: soft-margin tuning parameter of the SVM. Defaults to `10^c(-3:3)`.
- `stepsize`: amount of features that are discarded in each step of the feature elimination. Defaults to 10%.
- `useAllFeatures`: should all features be used for classification (see also `Details`).
- `mapping`: a mapping that defines how probe sets are summarized to genes.
- `Gsub`: an adjacency matrix that represents the underlying biological network.
- `d`: the damping factor which controls the influence of the network data and the fold change on the ranking of the genes. Defaults to 0.5.
fit.rrfe

**Value**

a RRFE fit object.

- `features`: the selected features
- `error.bound`: the span bound of the model
- `fit`: the fitted SVM model

**Note**


**Author(s)**

Marc Johannes <JohannesMarc@gmail.com>

**References**


**Examples**

```r
# Not run:
library(Biobase)
data(sample.ExpressionSet)
x <- t(exprs(sample.ExpressionSet))
y <- factor(pData(sample.ExpressionSet)$sex)
# create the mapping
library('hgu95av2.db')
mapped.probes <- mappedkeys(hgu95av2REFSEQ)
refseq <- as.list(hgu95av2REFSEQ[mapped.probes])
times <- sapply(refseq, length)
mapping <- data.frame(probesetID=rep(names(refseq), times=times), graphID=unlist(refseq), row.names=NULL, stringsAsFactors=FALSE)
mapping <- unique(mapping)
library(pathClass)
data(adjacency.matrix)
res.rrfe <- crossval(x, y, DEBUG=TRUE, theta.fit=fit.rrfe, folds=3, repeats=1, parallel=TRUE, Cs=10^(-3:3), mapping=mapping, Gsub=adjacency.matrix, d=1/2)

# End(Not run)
```
getGeneRanks  

*Calculate GeneRanks as used by RRFE*

**Description**

Uses the GeneRank to calculate the ranks for genes. Afterwards the ranks are transformed as needed for the RRFE algorithm.

**Usage**

```r
getGeneRanks(x, y, mapping, Gsub, method = "foldChange",
             d = 0.5)
```

**Arguments**

- `x`: a p x n matrix of expression measurements with p samples and n genes.
- `y`: a factor of length p comprising the class labels.
- `mapping`: a mapping that defines how probe sets are summarized to genes.
- `Gsub`: an adjacency matrix that represents the underlying biological network.
- `method`: see help of `summarizeProbes`
- `d`: the damping factor which controls the influence of the network data and the fold change on the ranking of the genes. Defaults to 0.5

**Value**

a ranking of the genes for which pathway knowledge was available.

**Author(s)**

Marc Johannes <JohannesMarc@gmail.com>

**References**


**Examples**

```r
# Not run:
library(pathClass)
data(example_data)
ranks = getGeneRanks(x, y, mapping=mapping, Gsub=adjacency.matrix)

# End(Not run)
```
**mapping**

A mapping of Refseq Protein IDs to probe set IDs for the gene expression data

**Description**

A mapping of the hgu95av2 probe set IDs in `x` to the Refseq protein IDs contained in `adjacency.matrix`.

**Author(s)**

Marc Johannes <johannesmarc@gmail.com>

**matchMatrices**

Matches the expression data to the adjacency matrix using the provided mapping.

**Description**

Usually the dimension of the graph and the expression data do not fit to each other. Additionally, most often the graph comprises another type of knowledge, i.e. the expression matrix measures 10,000 genes represented as 15,000 probe sets and the graph provides information on 7,000 proteins. Thus, a node (protein) of the graph might match to two probe sets in the expression matrix (since both target the gene encoding the protein). Therefore, this method uses the relationship between probe sets and i.e. proteins which is encoded in the mapping to create a graph of probe sets rather than a graph of proteins.

**Usage**

`matchMatrices(x, mapping, adjacency)`

**Arguments**

- `x` : the p x n expression matrix with p patients and n genes.
- `mapping` : a mapping which encodes the relationship between the colnames of `x` and the row/colnames of the adjacency matrix.
- `adjacency` : the adjacency matrix of the underlying graph structure.

**Value**

- the matched input
  - `x` : the expression matrix containing only the features which are also present in the adjacency matrix
  - `mapping` : the mapping containing only necessary information
  - `adjacency` : the adjacency matrix with the same number of nodes as features in `x`
Author(s)

Marc Johannes <JohannesMarc@gmail.com>

Examples

```r
## Not run:
library(Biocbase)
data(sample.ExpressionSet)
x <- t(exprs(sample.ExpressionSet))
y <- factor(pData(sample.ExpressionSet)$sex)
# create the mapping
library('hgu95av2.db')
mapped.probes <- mapped.keys(hgu95av2REFSEQ)
refseq <- as.list(hgu95av2REFSEQ[mapped.probes])
times <- sapply(refseq, length)
mapping <- data.frame(probesetID=rep(names(refseq), times=times), graphID=unlist(refseq),
row.names=NULL, stringsAsFactors=FALSE)
mapping <- unique(mapping)
library(pathClass)
data(adjacency.matrix)
matched <- matchMatrices(x=x, adjacency=adjacency.matrix, mapping=mapping)

## End(Not run)
```

---

plot.pathClassResult  *Prints the result of one or more cross-validation run(s)*

**Description**

This function creates boxplots of the distribution of AUC for each repeat of the cross-validation. In a second plot the ROC curve of the AUCs is shown. If your result contains more than one cross-validation result these are plotted one after the other.

**Usage**

```r
## S3 method for class 'pathClassResult'
plot(x, label = "", toFile = TRUE,
     fname = "Result", switchLabels = FALSE,
     avg = "horizontal", spread.estimate = "boxplot", ...)
```

**Arguments**

- `x` A result of crossval.
- `label` the main label of the plots.
- `toFile` Should the results plotted into PDF file(s). If your result contains more than one cross-validation one PDF file is created for each result.
- `fname` the name of the file to save the results in.
**predict.graphSVM**

If your AUC is below 0.5 you can switch the labels to get an AUC above 0.5.

The method for averaging the AUCs of several repeats. See `performance` for more information.

A method to show the variation around the average of the ROC curve. See `performance` for more information.

Currently ignored.

**Author(s)**

Marc Johannes <JohannesMarc@gmail.com>

**Examples**

```r
## Not run:
library(Biobase)
data(sample.ExpressionSet)
x <- t(exprs(sample.ExpressionSet))
y <- factor(pData(sample.ExpressionSet)$sex)
res.rfe <- crossval(x,y,DEBUG=TRUE,theta.fit=fit.rfe,folds=2,repeats=1,parallel=TRUE,Cs=10^(-3:3))
plot(res.rfe, toFile=FALSE)

## End(Not run)
```

---

**predict.graphSVM** *Predict Method for Graph-SVM Fits*

**Description**

Obtains predictions from a fitted graphSVM object.

**Usage**

```r
## S3 method for class 'graphSVM'
predict(object, newdata, type = "response", ...)
```

**Arguments**

- `object`: a fitted object of class inheriting from 'graphSVM'
- `newdata`: a matrix with variables to predict
- `type`: response gives the predictions class gives the predicted classes.
- `...`: currently ignored.

**Value**

The predictions.
Author(s)

Marc Johannes <JohannesMarc@gmail.com>

Examples

```r
## Not run:
library(pathClass)
data(example_data)
matched <- matchMatrices(x=x, adjacency=adjacency.matrix, mapping=mapping)
wk <- calc.diffusionKernel(L=matched$adjacency, is.adjacency=TRUE, beta=0) # beta should be tuned
fit <- fit.graph.svm(matched$x[1:5,], y[1:5], DEBUG=TRUE, mapping=matched$mapping, diffusionKernel=wk)
predict(fit, newdata=matched$x[6:10,])

## End(Not run)
```

drop

```
predict

Predict Method for Network-based SVM Fits

Description

Obtains predictions from a fitted `networkBasedSVM` object.

Usage

```
## S3 method for class 'networkBasedSVM'
predict(object, newdata, ...)
```

Arguments

- `object`: a fitted object of class inheriting from ‘networkBasedSVM’
- `newdata`: a matrix with variables to predict
- `...`: currently ignored.

Value

the predictions.

Author(s)

Marc Johannes <JohannesMarc@gmail.com>
Examples

```r
## Not run:
library(pathClass)
data(example_data)
matched <- matchMatrices(x=x, adjacency=adjacency.matrix, mapping=mapping)
ad.list <- as.adjacencyList(matched$adjacency)
fit = fit.networkBasedSVM(matched$x[1:5,], y[1:5], DEBUG=TRUE, adjacencyList=ad.list,
  lambdas=10^(-1:2), sd.cutoff=50)
predict(fit, newdata=matched$x[6:10,])

## End(Not run)
```

### predict.rfe  
**Predict Method for RFE Fits**

**Description**

Obtains predictions from a fitted RFE object.

**Usage**

```r
## S3 method for class 'rfe'
predict(object, newdata, type = "response", ...)
```

**Arguments**

- **object**: a fitted object of class inheriting from 'rfe'
- **newdata**: a matrix with variables to predict
- **type**: 
  - response: gives the predictions
  - class: gives the predicted classes.
  - ...: currently ignored.

**Value**

the predictions.

**Author(s)**

Marc Johannes <JohannesMarc@gmail.com>

**Examples**

```r
## Not run:
library(pathClass)
data(example_data)
fit = fit.rfe(x[1:5,], y[1:5], DEBUG=T)
predict(fit, newdata=x[6:10,])

## End(Not run)
```
predict.rrfe

Predict Method for RRFE Fits

Description

Obtains predictions from a fitted RRFE object.

Usage

```r
## S3 method for class 'rrfe'
predict(object, newdata, type = "response", ...)
```

Arguments

- `object`: a fitted object of class inheriting from 'rrfe'
- `newdata`: a matrix with variables to predict
- `type`: response gives the predictions class gives the predicted classes.
- `...`: currently ignored.

Value

the predictions.

Author(s)

Marc Johannes <JohannesMarc@gmail.com>

Examples

```r
## Not run:
library(pathClass)
data(example_data)
fit = fit.rrfe(x[1:5,], y[1:5], DEBUG=T, mapping=mapping, Gsub=adjacency.matrix)
predict(fit, newdata=x[6:10,])

## End(Not run)
```
read.hprd  
Parse the HPRD flat file

Description
This function parses the tab delimited flat file of protein-protein interactions coming from the HPRD (http://www.hprd.org/download).

Usage
read.hprd(fname, chipProteins = NULL)

Arguments
fname path to the HPRD flat file.
chipProteins limit the resulting adjacency matrix to certain proteins.

Value
An adjacency matrix

Author(s)
Marc Johannes <JohannesMarc@gmail.com>

Examples
## Not run:
hprd <- read.hprd('BINARY_PROTEIN_PROTEIN_INTERACTIONS.txt')
## End(Not run)

summarizeProbes  
Summarize probe sets

Description
Summarize multiple probe sets targeting one gene into one value for that gene. On most microarrays there will be more than one probe set for a gene. However, in the underlying network the gene will only be present one time. Therefore, in order to calculate a Gene(Page)Rank weight for this gene, all expression measurements have to be summarized.

Usage
summarizeProbes(exprs, mapping, method = "median",
                  groups = NULL, adjacency = NULL)
Arguments

exprs  \( n \times p \) matrix with \( n \) probe sets and \( p \) samples.

mapping a matrix or data.frame with 2 columns. The colnames of mapping have to contain at least 'graphID' and 'probesetID'. These two columns define the mapping between the probe sets on the microarray and the nodes of the graph.

method defines how several probe sets should be combined. One of median, mean, foldChange or none.

groups defines the grouping of samples. Only needed if method is foldChange.

adjacency a matrix that represents the graph of the underlying biological network.

Details

summarizes all probes of a gene to one value for that gene if the summarization method is 'none' then the only thing which is done is that all probesets for which no pathway is available are discarded.

Value

matrix with 1st column probeIDs 2nd column gene IDs

---

\( x \)  
Example gene expression data

Description

A data matrix \( x \) containing gene expression data of 10 patients

Author(s)

Marc Johannes <JohannesMarc@gmail.com>

---

\( y \)  
Example class labels for the gene expression data

Description

Class labels for the 10 patients contained the data matrix \( x \)

Author(s)

Marc Johannes <JohannesMarc@gmail.com>
Index

*Topic **datasets**
  adjacency.matrix, 3
  mapping, 15
  x, 22
  y, 22
*Topic **data**
  adjacency.matrix, 3
  mapping, 15
  x, 22
  y, 22
*Topic **package**
  pathClass-package, 2

adjacency.matrix, 3
as.adjacencyList, 3, 10

calc.diffusionKernel, 4, 8
crossval, 5

desummarize.ranks, 6

extractFeatures, 6

fit.graph.svm, 5, 7
fit.networkBasedSVM, 3, 5, 9
fit.rfe, 5, 11
fit.rrfe, 5, 12

geneRanks, 14

mapping, 15
matchMatrices, 15

pathClass (pathClass-package), 2
pathClass-package, 2
performance, 17
plot.pathClassResult, 16
predict.graphSVM, 17
predict.networkBasedSVM, 18
predict.rfe, 19
predict.rrfe, 20

read.hprd, 21
summarizeProbes, 14, 21
x, 22
y, 22