Package: geNetClassifier (via r-universe)

November 7, 2026

Type Package

Title Classify diseases and build associated gene networks using gene expression profiles

Version 1.52.0

Date 2020-04-02

Author Sara Aibar, Celia Fontanillo and Javier De Las Rivas. Bioinformatics and Functional Genomics Group. Cancer Research Center (CiC-IBMCC, CSIC/USAL). Salamanca. Spain.

Maintainer Sara Aibar <saibar@usal.es>

Depends R (>= 2.10.1), Biobase (>= 2.5.5), EBarrays, minet, methods

Imports e1071, graphics, grDevices

Suggests leukemiasEset, RUnit, BiocGenerics

Enhances RColorBrewer, igraph, infotheo

Description Comprehensive package to automatically train and validate a multi-class SVM classifier based on gene expression data. Provides transparent selection of gene markers, their coexpression networks, and an interface to query the classifier.

License GPL (>= 2)

ZipData no

URL http://www.cicancer.org

LazyLoad yes

Collate class.GenesRanking.R class.GenesNetwork.R class.GeneralizationError.R class.GeNetClassifierReturn.R functions.private.R functions.public.R function.main.R

biocViews Classification, DifferentialExpression, Microarray

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:37:36 UTC

RemoteUrl https://github.com/bioc/geNetClassifier

RemoteRef RELEASE_3_23

RemoteSha 59ec64e3abdc4a0f2ca2847a9458b5b80b2b9d3d

Contents

geNetClassifier-package . . . 2
calculateGenesRanking . . . 4
externalValidation.probMatrix . . . 7
externalValidation.stats . . . 8
gClasses-methods . . . 10
GeneralizationError-class . . . 10
genesDetails-methods . . . 12
GenesNetwork-class . . . 13
GenesRanking-class . . . 14
geneSymbols . . . 17
geNetClassifier . . . 18
GeNetClassifierReturn-class . . . 22
getEdges-methods . . . 25
getNodes-methods . . . 25
getNumEdges-methods . . . 26
getNumNodes-methods . . . 26
getRanking-methods . . . 27
getSubNetwork-methods . . . 27
getTopRanking-methods . . . 28
leukemiasClassifier . . . 29
network2txt . . . 30
numGenes-methods . . . 30
numSignificantGenes-methods . . . 31
overview-methods . . . 32
plot.GenesRanking . . . 33
plot.GeNetClassifierReturn . . . 34
plotAssignments . . . 35
plotDiscriminantPower . . . 37
plotExpressionProfiles . . . 39
plotNetwork . . . 41
queryGeNetClassifier . . . 43
querySummary . . . 45
setProperties-methods . . . 47

Index **48**

`geNetClassifier-package`

classify diseases and build associated gene networks using gene expression profiles

---

Description

Comprehensive package to automatically train a multi-class SVM classifier based on gene expression data. Provides transparent selection of gene markers, their coexpression networks, and an interface to query the classifier.

Details

Package:	geNetClassifier
Type:	Package
Version:	1.0
Date:	2013-02-28
License:	GPL (>=2)
LazyLoad:	yes
Depends:	R (>= 2.10.1), Biobase (>= 2.5.5), EBarrays, minet, methods
Imports:	e1071, ipred, graphics
Suggests:	leukemiasEset
Enhances:	RColorBrewer, igraph

Author(s)

Sara Aibar, Celia Fontanillo and Javier De Las Rivas
Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, CSIC/USAL). Salamanca. Spain.
Maintainer: Sara Aibar <saibar@usal.es>

See Also

Main functions included in this package:

- geNetClassifier
- queryGeNetClassifier

Query stats funtions:

- querySummary
- externalValidation.probMatrix
- externalValidation.stats
- plotAssignments

Plots and genes info:

- calculateGenesRanking
- plotNetwork
- plotDiscriminantPower
- plotExpressionProfiles

Classes:

- GenesRanking
- GenesNetwork
- GeNetClassifierReturn

- GeneralizationError

Related data sets:

- leukemiasEset

`calculateGenesRanking` *Calculate GenesRanking*

Description

Calculates the genes ranking and/or plots the posterior probability of the genes ordered by class ranking.

Usage

```
calculateGenesRanking(eset=NULL, sampleLabels=NULL,
numGenesPlot=1000, plotTitle="Significant genes", plotLp=TRUE,
lpThreshold = 0.95, numSignificantGenesType="ranked",
returnRanking="full", nullHiphothesisFilter=0.95,  nGenesExprDiff=1000,
geneLabels=NULL, precalcGenesRanking=NULL, IQRfilterPercentage= 0,
verbose=TRUE)
```

Arguments

`eset`	ExpressionSet or Matrix. Gene expression of the train samples (positive & non-logaritmic normalized values).
`sampleLabels`	Character. PhenoData variable (column name) containing the train samples class labels. Matrix or Factor. Class labels of the train samples.
`numGenesPlot`	Integer. Number of genes to plot.
`plotTitle`	Character. Plot title.
`plotLp`	Logical. If FALSE no plot is drawn.
`lpThreshold`	Numeric between 0 and 1. Required posterior probability value to consider a gene 'significant'.
`numSignificantGenesType`	Character. Type of count for number of genes over lpThreshold. • "global". Counts all genes of a class with posterior probability over lpThreshold, even if in the final ranking they were assigned to another class. • "ranked". Counts only genes assigned to each class.
`returnRanking`	Character. Type of ranking to return: • "full". Ranking of all available genes. • "lp"/"significant"/"lpThreshold"/TRUE. Ranking of the significant genes (genes with posterior probability over lpThreshold). • FALSE/NULL. No ranking is returned.

`nullHiphothesisFilter`
: Numeric between 0 and 1. Genes with a Null Hipothesis with a posterior probability over this threshold will be removed from the ranking.
Null Hipothesis: They don't represent any class.

`nGenesExprDiff`
: Numeric. Number of top genes to calculate the differencial expression for.

`geneLabels`
: Vector or Matrix. Gene name, ID or label which should be shown in the returned results and plots.

`IQRfilterPercentage`
: Integer. InterQuartile Range (IQR) filter applied to the initial data. Not recommended for more than two classes.

`precalcGenesRanking`
: Allows providing a `genesRanking` provided by `geNetClassifier` or by a previous execution for the same data and parameters.

`verbose`
: Logical. If TRUE, messages indicating the execution progress will be printed on screen.

Details

Significant genes: Genes with posterior probability over 'lpThreshold'.
More significant genes may mean:

- Very different class
- More systemic disease

Plot lines represet the posterior probability of genes, sorted by rank from left to right.

In order to find genes that diferentiate the classes from each other, the function ranks the genes bassed on their posterior probability for each class.
The posterior probability represents how well a gene differentiates samples from a class, from samples from other classes. Therefore, Genes with high posterior probability are good to differentiate a class from all the others.
This posterior probability is calculated by `emfit (pkg:EBarrays)`, an expectation-maximization (EM) algorithm for gene expression mixture model.

Value

- `GenesRanking` Optional. Requested genes ranking.
- Plot Optional. Plot of the posterior probability of the top genes.

See Also

`plot.GenesRanking` is a shortcut to plotting a previusly calculated genes ranking.
i.e. `plot(genesRanking)`

Examples

```
# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)

# Select the train samples:
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])

## Not run:
######
# Calculate/plot the significant genes (+ info) of a dataset
# without training classifier/calculating network
######
# Return only significant genes ranking (default)
signGenesRanking <- calculateGenesRanking(leukemiasEset[,trainSamples],
    sampleLabels="LeukemiaType")
numGenes(signGenesRanking)

# Return the full genes ranking:
fullRanking <- calculateGenesRanking(leukemiasEset[,trainSamples],
    sampleLabels="LeukemiaType", returnRanking="full")
numGenes(fullRanking)
numSignificantGenes(fullRanking)
# The significant genes can then be extracted from it:
signGenesRanking2  <- getTopRanking(fullRanking,
    numGenesClass=numSignificantGenes(fullRanking))
numGenes(signGenesRanking2)

# Changing the posterior probability required to consider genes significant:
signGenesRanking90 <- calculateGenesRanking(leukemiasEset[,trainSamples],
    sampleLabels="LeukemiaType", lpThreshold=0.9)
numGenes(signGenesRanking90)

## End(Not run)
######
# Ploting previously calculated rankings:
######
# Load or calculate a ranking (or a classifier with geNetClassifier)
data(leukemiasClassifier) # Sample trained classifier, @genesRanking

# Default plot:
# - equivalent to plot(leukemiasClassifier@genesRanking)
# - in this case, the previously calculated 'fullRanking'
#   is equivalent to 'leukemiasClassifier@genesRanking'
calculateGenesRanking(precalcGenesRanking=leukemiasClassifier@genesRanking)

# Changing arguments:
calculateGenesRanking(precalcGenesRanking=leukemiasClassifier@genesRanking,
    numGenesPlot=5000, plotTitle="Leukemias", lpThreshold=0.9)
```

`externalValidation.probMatrix` *Probability matrix.*

Description

Generates the probability matrix.

Usage

```
externalValidation.probMatrix(queryResult, realLabels, numDecimals=2)
```

Arguments

`queryResult`	Object returned by `queryGeNetClassifier`
`realLabels`	Factor. Actual/real class of the samples.
`numDecimals`	Integer. Number of decimals to return.

Details

A probability matrix contains the probabilities of assigning each assigned sample to each class. They help identifying where errors are likelly to occur even though there were not actual errors in the external/cross validation.

Value

The probability matrix.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

See Also

Main package function and classifier training: `geNetClassifier`
Query the classifier: `queryGeNetClassifier`
Query summary: `querySummary`
External validation stats: `externalValidation.stats`

Examples

```
#########################
## Classifier training
#########################

# Load an expressionSet:
library(leukemiasEset)
```

```
data(leukemiasEset)

# Select the train samples:
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])

# Train a classifier or load a trained one:
# leukemiasClassifier <- geNetClassifier(leukemiasEset[,trainSamples],
#    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier")
data(leukemiasClassifier) # Sample trained classifier

##########################
## External Validation
##########################
# Select the samples to query the classifier
#   - External validation: samples not used for training
testSamples <- c(1:60)[-trainSamples]

# Make a query to the classifier:
queryResult <- queryGeNetClassifier(leukemiasClassifier, leukemiasEset[,testSamples])

# Obtain the probability matrix for the assigned samples:
externalValidation.probMatrix(queryResult, leukemiasEset[,testSamples]$LeukemiaType)
```

`externalValidation.stats` *Statistics of the external validation.*

Description

Taking as input the confussion matrix resulting from external validation calculates the global Accuracy, Call Rate, Sensitivity, Specificity and Matthews Correlation Coefficient.

Usage

```
externalValidation.stats(confussionMatrix, numDecimals = 2)
```

Arguments

`confussionMatrix`	Confussion matrix containing the real class as rows and the assigned class as columns.
`numDecimals`	Integer. Number of decimals to show on the statistics.

Value

List:

- global General classifier stats.
  Accuracy: Percentage of correctly assigned samples within all assigned samples.
  CallRate: Percentage of samples wich were assigned to a class.
- byClass Stats by class.
  Sensitivity: Percentage of samples of each class which were correctly identified (Rate of true positives)
  Specificity: Percentage of samples assigned to a given class that really belonged to the class (Rate of true negatives)
  MCC (Matthews Correlation Coefficient): Measure wich takes into account both, true and false positives and negatives. (100%: Perfect assignments) confMatrix Confussion matrix.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

See Also

Main package function and classifier training: geNetClassifier
Querying the classifier: queryGeNetClassifier
Generating the probability matrix: externalValidation.probMatrix

Examples

```
#########################
## Classifier training
#########################

# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)

# Select the train samples:
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])

# Train a classifier or load a trained one:
# leukemiasClassifier <- geNetClassifier(leukemiasEset[,trainSamples],
#    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier")
data(leukemiasClassifier) # Sample trained classifier

#########################
## External Validation:
#########################
# Select the samples to query the classifier
#   - External validation: samples not used for training
testSamples <- c(1:60)[-trainSamples]

# Make a query to the classifier:
queryResult <- queryGeNetClassifier(leukemiasClassifier, leukemiasEset[,testSamples])
```

```
# Create the confusion matrix
confMatrix <- table(leukemiasEset[,testSamples]$LeukemiaType,queryResult$class)

# Calculate its accuracy, call rate, sensitivity and specificity:
externalValidation.stats(confMatrix)
```

`gClasses-methods` *Classes in the ranking.*

Description

Returns the names of the classes in a GenesRanking

Methods

`signature(object = "GenesRanking")`

See Also

Main package function and classifier training: `geNetClassifier`
This method's class (`GenesRanking`) help page.

Examples

```
data(leukemiasClassifier)
gClasses(leukemiasClassifier@genesRanking)
```

`GeneralizationError-class`
Class "GeneralizationError" (slot of GeNetClassifierReturn)

Description

Contains the estimation of the Generalization Error and the gene stats for `geNetClassifier` executed with the given data and parameters. \ Calculated by 5-fold cross-validation.

Slots

`accuracy`: `"Matrix"`. Accuracy and call rate.

`sensitivitySpecificity`: `"Matrix"`. Sensitivity, Specificity, Matthews Correlation Coefficient and Call Rate for each of the classes.

`confMatrix`: `"Matrix"`. Confussion matrix.

`probMatrix`: `"Matrix"`. Probabilities of belonging to each class for the assigned samples. Helps identifying where errors are likely to occur even though there were not actual errors in the cross-validation.

`querySummary`: "List". Stats regarding the probability and number of assigned test samples to each class.

`classificationGenes.stats`: "List". Some basic statistics regarding the chosen genes.

`classificationGenes.num`: "Matrix". Number of genes used for each of the 5 cross-validaton classifiers.

Methods

overview `signature(object = "GeneralizationError")`: Shows an overview of all the slots in the object.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

See Also

Main package function and classifier training: geNetClassifier

Examples

```
######
# Load data and train a classifier
######

# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)

# Select the train samples:
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])

# Train a classifier or load a trained one:
# Note: Required 'estimateGError=TRUE'
# leukemiasClassifier <- geNetClassifier(leukemiasEset[,trainSamples],
#    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier",
#    estimateGError=TRUE)
data(leukemiasClassifier) # Sample trained classifier

# Global view of the returned object and its structure:
leukemiasClassifier
names(leukemiasClassifier)

#########
# Exploring the cross validation stats
# Note: Required 'estimateGError=TRUE' in geNetClassifier()
#########
# Generalization Error estimated by cross-validation:
```

```
leukemiasClassifier@generalizationError
overview(leukemiasClassifier@generalizationError)
 # i.e. probabilityMatrix:
 leukemiasClassifier@generalizationError@probMatrix
 # i.e. statistics of the genes chosen in any of the CV loops for for AML:
 leukemiasClassifier@generalizationError@classificationGenes.stats$AML
```

---

genesDetails-methods *Details of the genes in the network.*

Description

Information of the genes in the ranking (table format).

Arguments

object	a GenesRanking
nGenes	integer. Number of genes to show per class
numDecimals	integer. Number of decimals to show in the numeric values
classes	character. Classes of the genes to show
genes	character. Genes to show

Value

A list containing a dataframe with the details of the genes of each class. For each gene, the following information is provided:

ranking	Ranking of the gene.
gERankMean	Mean rank the gene obtained in the cross-validation loops. Only available if geNetClassifier() was called with option estimateGError=TRUE (False by default).
class	Class the gene was chosen for (the class the gene differentiates from the other classes).
postProb	Posterior probability which the gene was assigned by the expectation-maximization algorithm (emfit). Tied values are ranked based on the higher absolute value of exprsMeanDiff. Values are rounded. Several genes may look tied at posterior probability '1' but may actually be i.e. 0.999998 and 0.999997.
exprsMeanDiff	Difference betwen the mean expression of the gene within its class and its mean expression in the other classes.
exprsUpDw	Gene repressed (DOWN) or over-expressed(UP) for the current class (compared to the other classes).
discriminantPower	Measure calculated based on the coordinates of the support vectors. Represents the weight that the classifier gives to each gene to separate the classes.
discrPwClass	Class for which the Discriminant Power was calculated for.
isRedundant	Does the gene have a high correlation or mutual information with other genes in the list? The threshold to consider a gene redundant can be set through the arguments (by default: correlationsThreshold=0.8 and interactionsThreshold=0.5).

Methods

genesDetails(object, nGenes=NULL, numDecimals=4, classes=NULL, genes=NULL)

See Also

Main package function and classifier training: geNetClassifier
This method's class (GenesRanking) help page.

Examples

```
data(leukemiasClassifier) # Sample geNetClassifier() return
options(width=200) # Optional, use in case the table rows are wrapped

genesDetails(leukemiasClassifier@classificationGenes)$CML
genesDetails(leukemiasClassifier@genesRanking, nGenes=5, numDecimals=2,
classes="AML")
genesDetails(leukemiasClassifier@genesRanking, genes=c("ENSG00000096006",
"ENSG00000168081","ENSG00000105699"))$CLL
```

GenesNetwork-class *Class "GenesNetwork"*

Description

Contains the network returned by geNetClassifier. (Slot: @genesNetwork)

Methods

getNodes signature(object = "GenesNetwork"): Returns the network nodes (genes).

getEdges signature(object = "GenesNetwork"): Returns the network edges (relationships).

getNumNodes signature(object = "GenesNetwork"): Returns the number of nodes (genes) in the network.

getNumEdges signature(object = "GenesNetwork"): Returns the number of edges (relationships) in the network,

getSubNetwork signature(network = "GenesNetwork"): Returns a new network containing only the given genes.

network2txt signature(network = "GenesNetwork"): Exports the network as text file.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

See Also

Main package function and classifier training: geNetClassifier Plot network or export as iGraph: plotNetwork

Examples

```
######
# Load data and train a classifier
######

# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)

# Select the train samples:
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])

# Train a classifier or load a trained one:
# leukemiasClassifier <- geNetClassifier(leukemiasEset[,trainSamples],
#    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier")
data(leukemiasClassifier) # Sample trained classifier

######
# Explore the returned object
######
# Global view of the object and its structure:
names(leukemiasClassifier)

# List of Networks by classes:
leukemiasClassifier@genesNetwork
# Access to the nodes or edges of each network:
getEdges(leukemiasClassifier@genesNetwork$AML)[1:5,]
getNodes(leukemiasClassifier@genesNetwork$AML)[1:50]


######
# Plotting
######
# Example: Plotting the sub-network of a class classificationGenes
# Get the sub-network containing only the classification genes:
subNet <- getSubNetwork(leukemiasClassifier@genesNetwork,
    leukemiasClassifier@classificationGenes)
# Get the classification genes' info/details:
clGenesInfo <- genesDetails(leukemiasClassifier@classificationGenes)

# Plot the network of the class "ALL"
plotNetwork(subNet$ALL, genesInfo=clGenesInfo,
    plotOnlyConnectedNodesNetwork=FALSE)
```

GenesRanking-class *Class "GenesRanking"*

Description

Contains a genes ranking and the genes info calculated by geNetClassifier.
(Slots @classificationGenes and @genesRanking from geNetClassifier output)

Methods

genesDetails signature(object = "GenesRanking"): Returns data.frames with information about the genes.

getRanking signature(object = "GenesRanking"): Returns a matrix containing the ranked genes.

getTopRanking signature(object = "GenesRanking", numGenesClass): Returns a new GenesRanking object containing only the top genes of each class.

gClasses signature(object = "GenesRanking"): Returns the classes for which the genes are ranked.

numGenes signature(object = "GenesRanking"): Returns the number of available ranked genes per class.

numSignificantGenes signature(object = "GenesRanking"): Returns the number of significant genes per class (genes over the given posterior probability threshold).

plot signature(x = "GenesRanking", y = "missing"): Plots the genes' posterior probability. Wrapper of calculateGenesRanking.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

See Also

For more information on how the ranking is calculated and how to interpret the given information, see the package vignette.

Main package function and classifier training: geNetClassifier
Plot the ranking genes's posterior probability: plot.GenesRanking

Examples

```
######
# Calculate a genesRanking
######

# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)

# Select the train samples:
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])

# Calculate the genesRanking with calculateGenesRanking()
```

```
## Not run:
genesRanking <- calculateGenesRanking(leukemiasEset[,trainSamples],
    sampleLabels="LeukemiaType", returnRanking="full")
## End(Not run)

# geNetClassifier() also calculates a genes ranking
# Sample output:
data(leukemiasClassifier)
genesRanking <- leukemiasClassifier@genesRanking

######
# Exploring the rankings
######
# Number of available genes in the ranking:
numGenes(genesRanking)

# Number of significant genes (genes with posterior probability over the threshold.
# Default: lpThreshold=0.95):
numSignificantGenes(genesRanking)

# Top 10 genes of CML:
genesDetails(genesRanking)$CML[1:10,]

# To get a sub ranking with the top 10 genes:
getTopRanking(genesRanking, 10)

# Genes details of the top 10 genes:
genesDetails(getTopRanking(genesRanking, 10))

######
# Exploring the genes used for training the classifier
######
numGenes(leukemiasClassifier@classificationGenes)
leukemiasClassifier@classificationGenes
#genesDetails(leukemiasClassifier@classificationGenes)  # List by classes
genesDetails(leukemiasClassifier@classificationGenes)$AML # Show a class genes
# If your R console wraps the table rows, try widening your display width:
# options(width=200)

######
# Creating a GenesRanking object
# i.e. To use geNetClassifier() with a ranking based on another algorithm
######

### 1. Calculate gene scores
# Two classes:
geneScore <- matrix(sample(seq(0,1,by=0.01), size=100, replace=TRUE))
colnames(geneScore) <- "BothClasses"
rownames(geneScore) <- paste("Gene", 1:100, sep="")

# More than two classes:
geneScore <- matrix(sample(seq(0,1,by=0.01), size=300, replace=TRUE), ncol=3)
colnames(geneScore) <- paste("Class", 1:3, sep="")
```

```
rownames(geneScore) <- paste("Gene", 1:100, sep="")

### 2. Create object
postProb <- geneScore
ord <- apply(postProb, 2, function(x) order(x, decreasing=TRUE))
numGenesClass <- apply(postProb, 2, function(x) sum(!is.na(x)))
customRanking <- new("GenesRanking", postProb=postProb, ord=ord, numGenesClass=numGenesClass)

# GenesRanking object ready:
customRanking
genesDetails(customRanking)
customRanking@numGenesClass
numSignificantGenes(customRanking)

# geNetClassifier(..., precalcGenesRanking = customRanking)
```

`geneSymbols` *Gene symbols associated to human Ensemble IDs.*

Description

Gene symbols to use as gene labels in the package examples.

Source: simplified version of genes.human.annotation from GATExplorer (http://bioinfow.dep.usal.es/xgate/mapping/mapping.php?content=annotationfiles).

Usage

```
data(geneSymbols)
```

Format

Named character vector containing the gene symbol as content, and the associated Ensemble ID as name.

Examples

```
data(geneSymbols)
head(geneSymbols)
```

geNetClassifier	*Main function of the geNetClassifier package.* *Trains the multi-class SVM classifier based on the given gene expression data through transparent detection of gene markers and their associated networks.*

Description

Allows to train the classifier, calculate the genes network...

Usage

```
geNetClassifier(eset, sampleLabels, plotsName = NULL,
buildClassifier = TRUE, estimateGError = FALSE,
calculateNetwork = TRUE, labelsOrder = NULL, geneLabels = NULL,
numGenesNetworkPlot = 100,
minGenesTrain = 1, maxGenesTrain = 100, continueZeroError = FALSE,
numIters = 6, lpThreshold = 0.95, numDecimals = 3,
removeCorrelations = FALSE, correlationsThreshold = 0.8,
correlationMethod = "pearson",
removeInteractions = FALSE, interactionsThreshold = 0.5,
minProbAssignCoeff = 1, minDiffAssignCoeff = 0.8,
IQRfilterPercentage = 0, skipInteractions = TRUE,
precalcGenesNetwork = NULL, precalcGenesRanking = NULL,
returnAllGenesRanking = TRUE, kernel="linear", verbose=TRUE, ...)
```

Arguments

eset	ExpressionSet or matrix. Gene expression of the train samples (positive & non-logaritmic normalized values).
sampleLabels	Character. PhenoData variable (column name) containing the train samples class labels. Matrix or Factor. Class labels of the train samples.
labelsOrder	Vector or Factor. Order in which the labels should be shown in the returned results and plots.
plotsName	Character. File name with which the plots should be saved. If not provided, no plots will be drawn.
buildClassifier	Logical. If TRUE trains a classifier with the given samples.
estimateGError	Logical. If TRUE uses cross-validation to estimate the Generalization Error of a classiffier trained with the given samples.
calculateNetwork	Logical. If TRUE calculates the coexpression network between the best genes.
geneLabels	Vector or Matrix. Gene name, ID or label which should be shown in the returned results and plots.

`numGenesNetworkPlot`
: Integer. Number of genes to show in the coexpression network for each class.

`minGenesTrain`
: Integer. Maximum number of genes per class to train the classifier with.

`maxGenesTrain`
: Integer. Maximum number of genes per class to train the classifier with.

`continueZeroError`
: Logical. If TRUE, the program will continue testing combinations with more genes even if error 0 has been reached.

`numIters`
: Integer. Number of iterations to determine the optimum number of genes (between `minGenesTrain` and `maxGenesTrain`).

`lpThreshold`
: Numeric between 0 and 1. Required posterior probability value to consider a gene 'significant'.

`removeCorrelations`
: Logical. If TRUE, no correlated genes will be chosen to train the classifier.

`correlationsThreshold`
: Numeric between 0 and 1. Minimum Pearson's correlation coefficient to consider genes correlated.

`correlationMethod`
: "pearson", "kendall" or "spearman". Type of correlation to calculate between genes.

`removeInteractions`
: Logical. If TRUE, genes with Mutual Information coefficient over the threshold will not be chosen to train the classifier.

`interactionsThreshold`
: Numeric between 0 and 1. Minimum Mutual Information coefficient to consider two genes equivalent.

`numDecimals`
: Integer. Number of decimals to show in the statistics.

`minProbAssignCoeff`
: Numeric. Allows modifying the required probability to assign a sample to a class in the internal crossvalidation. For details see: `queryGeNetClassifier`

`minDiffAssignCoeff`
: Numeric. Allows modifying the difference of probabilities required between the most likely class and second most likely class to assign a sample. For details see: `queryGeNetClassifier`

`IQRfilterPercentage`
: Integer. InterQuartile Range (IQR) filter applied to the initial data. Not recommended for more than two classes.

`skipInteractions`
: Logical. If TRUE, the interactions between genes are not calculated (they will not appear on the genes network). Saves some execution time. Only available if `removeInteractions=FALSE`.

`precalcGenesNetwork`
: `GenesNetwork` from a previous execution with the same expression data and parameters.

`precalcGenesRanking`
: `GenesRanking` from a previous execution with the same expression data and parameters.

`returnAllGenesRanking`
: Logical. If TRUE, returns the whole genes ranking. If FALSE the returned ranking contains only the significant genes (genes over lpThreshold).

`verbose`
: Logical. If TRUE, messages indicating the execution progress will be shown.

`kernel`
: Character. Type of SVM kernel. Default: "linear",

`...`
: Other arguments to pass to the `svm` function.

Value

A `GeNetClassifierReturn` object containing the classifier and the genes chosen to train it (`classificationGenes`), Cross-Validation statistics, the whole `GenesRanking` and each class' `GenesNetwork` (if requested).
Several plots saved as '`plotsName_....pdf`' in the working directory.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

References

Packages used by this function:
`EBarrays: emfit` (Implements EM algorithm for gene expression mixture model) and ebPatterns, for calculating the gene ranking.
Ming Yuan, Michael Newton, Deepayan Sarkar and Christina Kendziorski (2007). EBarrays: Unified Approach for Simultaneous Gene Clustering and Differential Expression Identification. R package.

`e1071: svm`.
Evgenia Dimitriadou, Kurt Hornik, Friedrich Leisch, David Meyer and Andreas Weingessel (2011). e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. R package. http://CRAN.R-project.org/package=e1071

`ipred: kfoldcv` (computes feasible sample sizes for the k groups in k-fold cv) for the cross-validations.
Andrea Peters and Torsten Hothorn (2012). ipred: Improved Predictors. R package. http://CRAN.R-project.org/package=ipred

`minet` for the Mutual Information network.
Patrick E. Meyer, Frederic Lafitte and Gianluca Bontempi (2008). MINET: An open source R/Bioconductor Package for Mutual Information based Network Inference. BMC Bioinformatics. http://www.biomedcentral.com/1471-2105/9/461

`RColorBrewer` (`brewer.pal`) for palettes in some of the plots.
Erich Neuwirth (2011). RColorBrewer: ColorBrewer palettes. R package. http://CRAN.R-project.org/package=RColorBrewer

`igraph` for the graphical representation of the networks.
Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal,

Complex Systems 1695. 2006. http://igraph.sf.net

See Also

To query the classifier: queryGeNetClassifier
All functions in the package: geNetClassifier-package

Examples

```
########
# Load libraries and training data
########

# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)

# Select the train samples:
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])


########
# Training
########

# NOTE: Training the classifier takes a while...
# Choose ONE of the followings, or modify to suit your needs:
## Not run:

# "Basic" execution: All default parameters
leukemiasClassifier <- geNetClassifier(eset=leukemiasEset[,trainSamples],
    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier")

# All default parameters also estimatings the classiffier's Generalization Error:
# ( by default:  buildClassifier=TRUE, calculateNetwork=TRUE)
# Takes longer time than the basic execution
leukemiasClassifier <- geNetClassifier(eset=leukemiasEset[,trainSamples],
    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier",
    estimateGError=TRUE)

# Faster execution (few minutes - depending on the computer):
# By skipping the calculation of the interactions (MI) betwen the genes,
# and reducing the number of genes to explore when training the classifier
# (100 by default), the execution time can be sightly reduced
leukemiasClassifier <- geNetClassifier(eset=leukemiasEset[,trainSamples],
sampleLabels="LeukemiaType", plotsName="leukemiasClassifier",
skipInteractions= TRUE, maxGenesTrain=20)

# To any of these examples, you can add/remove the argument geneLabels,
# in order to show/remove the gene name in the rankings and plots:
```

```
# The argument labelsOrder allows showing the classes in a specific order
# i.e.: labelsOrder=c("ALL","CLL","AML",CML","NoL")

save(leukemiasClassifier, file="leukemiasClassifier.RData")  # Save execution result
# For loading the saved object in the future...
# (If it doesn't find it, use getwd() to make sure you are in the right directory)
#load("leukemiasClassifier.RData")


# To avoid having to train a classifier to continue learning to use the package,
# you can load the package's pre-executed example:
data(leukemiasClassifier)
#This example classifier was trained with the following code:
#leukemiasClassifier <- geNetClassifier(leukemiasEset[,trainSamples],
#    "LeukemiaType", plotsName="leukemiasClassifier", buildClassifier=TRUE,
#    estimateGError=TRUE, calculateNetwork=TRUE, geneLabels=geneSymbols)

########
# Explore the returned object:
########
names(leukemiasClassifier)
# More details on the class' help file:
?GeNetClassifierReturn

# Further options:
# The trained classifier can be used to find the class of new samples:
?queryGeNetClassifier

# The default plots can be modified and presonalized to fit the user needs:
?calculateGenesRanking
?plotNetwork
?plotDiscriminantPower
?plotExpressionProfiles

## End(Not run)
```

`GeNetClassifierReturn-class`
Class "GeNetClassifierReturn"

Description

Object wich wraps all the items returned by `geNetClassifier`. It usually contains the classifier, the genes ranking and information, the network and any other requested statistics.

Methods

names `signature(x = "GeNetClassifierReturn")`: Shows the available slots in the object.

overview `signature(object = "GeNetClassifierReturn")`: Shows an overview of all the slots in the object.

Slots

Available slots deppends on the arguments used to call `geNetClassifier()`:

`call`: call. Always available.

`classifier`: list. SVM classifier. Only available if geNetClassifier() was called with option `buildClassifier=TRUE` (default settings).

`classificationGenes`: `GenesRanking`. Genes used to train the classifier. Only available if geNetClassifier() was called with option `buildClassifier=TRUE` (default settings).

`generalizationError`: `GeneralizationError`. Statistics calculated for the current training set and options.
Only available if geNetClassifier() was called with option `estimateGError=TRUE` (False by default).

`genesRanking`: `GenesRanking`. Whole genes ranking (if `returnAllGenesRanking=TRUE`) or significant genes ranking (if `returnAllGenesRanking=FALSE`, includes only the genes with posterior probability over `lpThreshold`)

`genesRankingType`: character. "all", "significant" or "significantNonRedundant"

`genesNetwork`: List of `GenesNetwork`. Only available if geNetClassifier() was called with option `calculateNetwork=TRUE` (default settings).

`genesNetworkType`: character. At the moment, only "topGenes" available.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

See Also

Main package function and classifier training: `geNetClassifier`
`plot.GeNetClassifierReturn`

Examples

```
######
# Load data and train a classifier
######

# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)

# Select the train samples:
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])
```

```
# Train a classifier or load a trained one:
# leukemiasClassifier <- geNetClassifier(leukemiasEset[,trainSamples],
#    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier")
data(leukemiasClassifier) # Sample trained classifier

######
# Explore the returned object
######
# Global view of the object and its structure:
leukemiasClassifier
names(leukemiasClassifier)

### Depending on the available slots:
# Call and acess to the classifier:
leukemiasClassifier@call
leukemiasClassifier@classifier

# Genes used for training the classifier:
numGenes(leukemiasClassifier@classificationGenes)
leukemiasClassifier@classificationGenes
# Show de tetails of the genes of a class
genesDetails(leukemiasClassifier@classificationGenes)$AML
# If your R console wraps the table rows, try widening your display width:
# options(width=200)

# Generalization Error estimated by cross-validation:
leukemiasClassifier@generalizationError
overview(leukemiasClassifier@generalizationError)
# i.e. probabilityMatrix:
leukemiasClassifier@generalizationError@probMatrix
# i.e. statistics of the genes chosen in any of the CV loops for for AML:
leukemiasClassifier@generalizationError@classificationGenes.stats$AML

# List of Networks by classes:
leukemiasClassifier@genesNetwork
# Access to the nodes or edges of each network:
 getEdges(leukemiasClassifier@genesNetwork$AML)
 getNodes(leukemiasClassifier@genesNetwork$AML)

# Genes ranking:
leukemiasClassifier@genesRanking
 # Number of available genes in the ranking:
 numGenes(leukemiasClassifier@genesRanking)
 # Number of significant genes
 # (genes with posterior probability over lpThreshold, default=0.95)
 numSignificantGenes(leukemiasClassifier@genesRanking)
 # Top 10 genes of CML:
 genesDetails(leukemiasClassifier@genesRanking)$CML[1:10,]
 # To get a sub ranking with the top 10 genes:
 getTopRanking(leukemiasClassifier@genesRanking, 10)
 # Genes details of the top 10 genes:
 genesDetails(getTopRanking(leukemiasClassifier@genesRanking, 10))
```

getEdges-methods *Edges in the network.*

Description

Returns the network's edges (relations between genes).

Methods

`signature(object = "GenesNetwork")`

See Also

Main package function and classifier training: `geNetClassifier`
This method's class (`GenesNetwork`) help page.

Examples

```
data(leukemiasClassifier)
getEdges(leukemiasClassifier@genesNetwork$AML)[1:5,]
```

getNodes-methods *Nodes in the network.*

Description

Returns the network's nodes (genes).

Methods

`signature(object = "GenesNetwork")`

See Also

Main package function and classifier training: `geNetClassifier`
This method's class (`GenesNetwork`) help page.

Examples

```
data(leukemiasClassifier)
getNodes(leukemiasClassifier@genesNetwork$AML)[1:5]
```

getNumEdges-methods *Number of edges in the network.*

Description

Returns the number of edges (relationships) in the network.

Methods

`signature(object = "GenesNetwork")`

See Also

Main package function and classifier training: geNetClassifier
This method's class (GenesNetwork) help page.

Examples

```
data(leukemiasClassifier)
getNumEdges(leukemiasClassifier@genesNetwork$AML)
```

getNumNodes-methods *Number of nodes in the network.*

Description

Returns the number of nodes (genes) in the network.

Methods

`signature(object = "GenesNetwork")`

See Also

Main package function and classifier training: geNetClassifier
This method's class (GenesNetwork) help page.

Examples

```
data(leukemiasClassifier)
getNumNodes(leukemiasClassifier@genesNetwork$AML)
```

getRanking-methods *Shows the genes ranking.*

Description

Shows the ranking as matrix: Ranked genes by classes.

Arguments

`object`	a GenesRanking
`showGeneID`	boolean. If TRUE, the genes will be shown with the gene IDs used in the expressionSet. This matrix will be `...$geneID` in the returned list.
`showGeneLabels`	boolean. If TRUE, and if the ranking contains gene labels, the ranking matrix will use them. This matrix will be `...$geneLabels` in the returned list.

Value

The method returns a list with one or two matrices: `...$geneLabels` and `...$geneID`.

See Also

Main package function and classifier training: `geNetClassifier`
This method's class (`GenesRanking`) help page.

Examples

```
data(leukemiasClassifier)
getRanking(leukemiasClassifier@classificationGenes)

# Top 7 genes (two ways):
getRanking(leukemiasClassifier@genesRanking)$geneLabels[1:7,]
getRanking(getTopRanking(leukemiasClassifier@genesRanking, 7))

# Show gene ID and select a class:
getRanking(leukemiasClassifier@classificationGenes, showGeneID=TRUE
)$geneID[,"CML", drop=FALSE]
```

getSubNetwork-methods *Get a sub-network.*

Description

Returns the sub-network formed by the given genes.

Usage

```
getSubNetwork(network, genes, showWarnings=TRUE)
```

Arguments

network	GenesNetwork or GenesNetwork list containing the whole network.
genes	GenesRanking or character vector. Genes in the new network.
showWarnings	Logical. If true, shows warnings if the given genes are not in the network.

Value

A GenesNetwork or list of networks between the given genes.

See Also

Main package function and classifier training: geNetClassifier
This method's class (GenesNetwork) help page.

Examples

```
data(leukemiasClassifier)
clGenesSubNet <- getSubNetwork(leukemiasClassifier@genesNetwork,
leukemiasClassifier@classificationGenes)
getSubNetwork(leukemiasClassifier@genesNetwork, getTopRanking(leukemiasClassifier@genesRanking, numGenesClass=3
```

---

getTopRanking-methods *Gets a new ranking with the given top genes.*

Description

Returns a new ranking containing only the top genes of each class.

Arguments

object	a GenesRanking
numGenesClass	integer. Number of genes per class.

Methods

```
getTopRanking(object, numGenesClass)
```

See Also

Main package function and classifier training: geNetClassifier
This method's class (GenesRanking) help page.

Examples

```
data(leukemiasClassifier) # Sample classifier

# Sub-ranking with the top 10 genes:
getTopRanking(leukemiasClassifier@genesRanking, 10)
```

leukemiasClassifier *Sample leukemias classifier*

Description

A sample of the object returned by geNetClassifier. Containins the classifier, the network, and the gene statistics.

Usage

```
data(leukemiasClassifier)
```

Format

GeNetClassifierReturn object

Examples

```
data(leukemiasClassifier)
# Global view of the object and its structure:
leukemiasClassifier
names(leukemiasClassifier)

# Call and acess to the classifier:
leukemiasClassifier@call
leukemiasClassifier@classifier

# Genes used for training the classifier:
numGenes(leukemiasClassifier@classificationGenes)
leukemiasClassifier@classificationGenes
genesDetails(leukemiasClassifier@classificationGenes)

# Generalization Error estimated by cross-validation:
#  leukemiasClassifier@generalizationError
# overview(leukemiasClassifier@generalizationError)

# List of Networks by classes:
leukemiasClassifier@genesNetwork

# Access to the nodes or edges of each network:
getEdges(leukemiasClassifier@genesNetwork$AML)[1:5,]
getNodes(leukemiasClassifier@genesNetwork$AML)[1:50]

# Global genes ranking:
leukemiasClassifier@genesRanking
numGenes(leukemiasClassifier@genesRanking)
numSignificantGenes(leukemiasClassifier@genesRanking)
# getTopRanking(leukemiasClassifier@genesRanking, 10)
```

network2txt *network2txt*

Description

Saves the GenesNetwork as text file.

Usage

```
network2txt(network, filePrefix = NULL, nwClass = NULL)
```

Arguments

network	GenesNetwork or list of GenesNetworks.
filePrefix	Character. File name prefix.
nwClass	Character. Network class.

Value

Saves the networks as text (.txt) files. The files will be saved in the current working directory as filePrefix_*className*.txt.

See Also

Main package function and classifier training: geNetClassifier
This method's class (GenesNetwork) help page.

Examples

```
## Load or calculate a network:

data(leukemiasClassifier)

## Export as text:
network2txt(leukemiasClassifier@genesNetwork, filePrefix="leukemiasNetwork")
```

numGenes-methods *Number of genes in the genesRanking.*

Description

Provides the number of genes in the genesRanking.

Methods

```
signature(object = "GenesRanking")
```

See Also

Main package function and classifier training: geNetClassifier
This method's class (GenesRanking) help page.

Examples

```
data(leukemiasClassifier)
numGenes(leukemiasClassifier@genesRanking)
```

---

numSignificantGenes-methods
Number of ranked genes over the posterior probability threshold.

Description

Provides the number of ranked genes over the posterior probability threshold

Arguments

object	a GenesRanking
lpThreshold	Posterior probability threshold
numSignificantGenesType	"ranked" or "global". Ranked will show the count of genes on the ranking of each class. Each gene will be counted only once, since it is only kept in the class for which it had better ranking. Global counts the genes over the threshold before assigning them to a class. i.e. a gene might have 0.3 for one class, and 0.25 for another, if we are taking a thershold of 0.20, it will be counted on both classes.

Methods

```
numSignificantGenes(object, lpThreshold=0.95, numSignificantGenesType="ranked")
```

See Also

Main package function and classifier training: geNetClassifier
This method's class (GenesRanking) help page.

Examples

```
data(leukemiasClassifier)
# Total number of genes in the ranking:
numGenes(leukemiasClassifier@genesRanking)
# Number of genes ovher the posterior probability threshold
# Default:  lpThreshold=0.95, numSignificantGenesType="ranked"
```

```
numSignificantGenes(leukemiasClassifier@genesRanking)
numSignificantGenes(leukemiasClassifier@genesRanking, numSignificantGenesType="global")
numSignificantGenes(leukemiasClassifier@genesRanking, lpThreshold=0.90)
```

`overview-methods` *Overview*

Description

Provides an overview of all the slots in the object.

Methods

It can be applied to the following classes:

`signature(object = "GenesNetwork")`

`signature(object = "GenesRanking")`

`signature(object = "GeNetClassifierReturn")`

`signature(object = "GeneralizationError")`

See Also

Main package function and classifier training: geNetClassifier
This method's classes help pages:
GenesRanking
GenesNetwork
GeNetClassifierReturn
GeneralizationError

Examples

```
data(leukemiasClassifier)
# geNetClassifier return:
overview(leukemiasClassifier)
# Generalization Error and stats estimated by cross-validation:
overview(leukemiasClassifier@generalizationError)
# A GenesNetwork:
# (a class has to be selected, otherwise it is a list)
overview(leukemiasClassifier@genesNetwork$ALL)

# For a GenesRanking,  we recommend to use genesDetails() instead:
genesDetails(leukemiasClassifier@classificationGenes)$AML
```

`plot.GenesRanking` *Plot GenesRanking*

Description

Plots the posterior probability of the genes ordered by class ranking.

Usage

```
## S3 method for class 'GenesRanking'
plot(x, y="missing", numGenesPlot=1000,
    plotTitle="Significant genes", lpThreshold = 0.95, ...)
```

Arguments

`x`	GenesRanking.
`numGenesPlot`	Numeric. Number of genes to plot.
`plotTitle`	Character. Plot main title.
`lpThreshold`	Numeric between 0 and 1. Required posterior probability value to consider a gene 'significant'.
`y`	Not required.
`...`	Not required

Details

Significant genes: Genes with posterior probability over 'lpThreshold'.
More significant genes may mean:

- Very different class
- More systemic disease

Plot lines represet the posterior probability of genes, sorted by rank from left to right.

In order to find genes that diferentiate the classes from each other, the function ranks the genes bassed on their posterior probability for each class.
The posterior probability represents how well a gene differentiates samples from a class, from samples from other classes. Therefore, Genes with high posterior probability are good to differentiate a class from all the others.
This posterior probability is calculated by `emfit (pkg:EBarrays)`, an expectation-maximization (EM) algorithm for gene expression mixture model.

Value

Posterior probability plot of the top genes.

Examples

```
# Load or calculate a ranking (or a classifier with geNetClassifier)
data(leukemiasClassifier) # Sample trained classifier, @genesRanking

# Default plot:
plot(leukemiasClassifier@genesRanking)

# Changing options:
plot(leukemiasClassifier@genesRanking,
    numGenesPlot=5000, plotTitle="Leukemias", lpThreshold=0.9)
```

`plot.GeNetClassifierReturn`

Plot GeNetClassifierReturn

Description

Allows generating the plots from the objet created by `geNetClassifier`.

Usage

```
## S3 method for class 'GeNetClassifierReturn'
plot(x, y="missing", fileName = NULL, lpThreshold = 0.95,
    numGenesLpPlot = 1000, numGenesNetworkPlot = 100,
    geneLabels = NULL, verbose = TRUE, ...)
```

Arguments

`x`	GeNetClassifierReturn. Object returned by the main function "`geNetClassifier`".
`fileName`	Character. File name to save the plots.
`lpThreshold`	Numeric between 0 and 1. Required posterior probability value to consider a gene 'significant'.
`numGenesLpPlot`	Integer. Number of genes to show in the significant genes plot.
`numGenesNetworkPlot`	Integer. Number of genes (nodes) to plot in the network.
`geneLabels`	Vector or Matrix. Gene name, ID or label which should be shown in the returned results and plots.
`verbose`	Logical. If TRUE, messages indicating the execution progress will be printed on screen.
`y`	Not required.
`...`	Not required

Details

The plots are generated by default by `geNetClassifier`. This function allows re-plotting them with different parameters.

Value

Plots (depending on the available info):
- Significant genes
- Classification genes' Discriminant Power
- Top ranked genes network (for each class)

See Also

Main package function and classifier training: geNetClassifier
Class GeNetClassifierReturn
Other plotting functions:
- plotDiscriminantPower
- plot.GenesRanking
- plotNetwork

Examples

```
# Train or load an already trained classifier
data(leukemiasClassifier)

# Plot default plots on-screen
plot(leukemiasClassifier)

# Save plots on file
# (includes Discriminant Power of all genes, but the networks will not be interactive)
plot(leukemiasClassifier, fileName="newPlots")
```

---

plotAssignments *Plot assignment probabilities*

Description

Plots the assignment probabilities of a previous query.

Usage

```
plotAssignments(queryResult, realLabels,
  minProbAssignCoeff = 1, minDiffAssignCoeff = 0.8,
  totalNumberOfClasses = NULL, pointSize=0.8, identify = FALSE)
```

Arguments

queryResult	Object returned by queryGeNetClassifier
realLabels	Factor. Actual/real class of the samples.
minProbAssignCoeff	Numeric. See queryGeNetClassifier for details.
minDiffAssignCoeff	Numeric. See queryGeNetClassifier for details.
totalNumberOfClasses	Numeric. Total number of classes the classifier was trained with. The assignment probability is determined bassed on it. It is not needed if there are samples of all the training classes.
pointSize	Numeric. Point size modifier.
identify	Logical. If TRUE and supported (X11 or quartz devices), the plot will be interactive and clicking on a point will identify the sample the point represents. Press ESC or right-click on the plot screen to exit.

Value

Plot.

See Also

Main package function and classifier training: geNetClassifier
Querying the classifier: queryGeNetClassifier

Examples

```
##########################
## Classifier training
##########################

# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)

# Select the train samples:
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])

# Train a classifier or load a trained one:
# leukemiasClassifier <- geNetClassifier(leukemiasEset[,trainSamples],
#    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier")
data(leukemiasClassifier) # Sample trained classifier

##########################
## External Validation:
##########################
# Select the samples to query the classifier
```

```
#   - External validation: samples not used for training
testSamples <- c(1:60)[-trainSamples]

# Make a query to the classifier:
queryResult <- queryGeNetClassifier(leukemiasClassifier, leukemiasEset[,testSamples])

#########################
## Plot:
#########################
plotAssignments(queryResult, realLabels=leukemiasEset[,testSamples]$LeukemiaType)
```

`plotDiscriminantPower` *Plots the genes' Discriminant Power.*

Description

Calculates and plots the Discriminant Power of the genes in the given classifier.

Usage

```
plotDiscriminantPower(classifier, classificationGenes = NULL,
geneLabels = NULL, classNames = NULL, plotDP = TRUE,
fileName = NULL, returnTable = FALSE, verbose = TRUE)
```

Arguments

`classifier`	Classifier returned by `geNetClassifier`. (@classifier)
`classificationGenes`	Vector or Matrix. IDs of the genes to plot. If matrix: genes should be ordered by classes. Columns should be named after the classes.
`geneLabels`	Vector or Matrix. Gene name, ID or label which should be shown in the returned results and plots.
`classNames`	Named vector. Short version of the class names if different from the ones used to train the classifier.
`plotDP`	Logical. If TRUE, plots the discriminant power of the given genes.
`fileName`	Character. File name to save the plot with. If not provided, the plots will be shown through the standard output device.
`returnTable`	Logical. If TRUE, returns a table with the genes discriminant power.
`verbose`	Logical. If TRUE, messages indicating the execution progress will be printed on screen.

Details

The Discriminant Power represents the weight the (SVM) classifier gives each gene to separate the classes. It is calculated based on the coordinates of the support vectors. Genes with a high Discriminant Power are better for identifying samples from the class.

Value

- Data frame Optional. Data.frame containing the genes and their Discriminant Power.
- Discriminant Power plot Optional. Shown throught the standard output devide or saved in the working directory as 'fileName.pdf' if `fileName` was provided.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

See Also

Main package function and classifier training: `geNetClassifier`

Examples

```
######
# Load data and train a classifier
######

# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)

# Select the train samples:
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])

# Train a classifier or load a trained one:
# leukemiasClassifier <- geNetClassifier(leukemiasEset[,trainSamples],
#    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier")
data(leukemiasClassifier) # Sample trained classifier

######
# Discriminant Power
######
# Default (plots up to 20 genes)
plotDiscriminantPower(leukemiasClassifier)
# Plot a specific gene:
plotDiscriminantPower(leukemiasClassifier, classificationGenes="ENSG00000169575")
# Plot top5 genes of a class, and return their discriminant power:
# Note: The discriminant Power can only be calculated for 'classificationGenes'
#           (genes chosen for training the classifier)
genes <- getRanking(leukemiasClassifier@classificationGenes,
    showGeneID=TRUE)$geneID[1:5,"AML",drop=FALSE] # Top 5 genes of AML
discPowerTable2 <- plotDiscriminantPower(leukemiasClassifier,
    classificationGenes=genes, returnTable=TRUE)

# For plotting more than 20 genes or saving the plots as .pdf, provide a fileName
plotDiscriminantPower(leukemiasClassifier,
     fileName="leukemiasClassifier_DiscriminantPower.pdf")
```

plotExpressionProfiles *Expression profiles plot.*

Description

Plots the expression profiles of the given genes.

Usage

```
plotExpressionProfiles(eset, genes=NULL, fileName=NULL,
geneLabels=NULL, type="lines", sampleLabels=NULL, sampleColors=NULL,
labelsOrder=NULL, classColors=NULL, sameScale=TRUE,
showSampleNames=FALSE, showMean= FALSE, identify=TRUE, verbose=TRUE)
```

Arguments

eset	ExpressionSet or Matrix. Gene expression of the samples.
genes	Vector or Matrix. IDs of the genes to plot. If matrix: genes should be ordered by classes. Columns should be named after the classes. If not provided, all available genes will be plot. Warning: If a list of genes is not provided, it will plot all available genes.
fileName	Character. File name to save the plots. If not provided, up to 20 genes will be shown on screen.
geneLabels	Vector or Matrix. Gene name, ID or label which should be shown in the returned results and plots.
type	Character. Plot type: "lines" or "boxplot".
sampleLabels	Character. PhenoData variable (column name) containing the train samples class labels. Matrix or Factor. Class labels of the train samples.
sampleColors	Character. Colors for the lines of the samples.
labelsOrder	Vector or Factor. Order in which the labels should be shown in the returned results and plots.
classColors	Character. Colors for each of the classes or samples of the class. Provide either sampleColors or classColors, not both.
sameScale	Logical. If TRUE, plots all the genes in the same expression scale.
showSampleNames	Logical. If TRUE, the sample names are shown in the plot. Not recommended for big datasets.
showMean	Logical. If TRUE, plots the class expression mean.
identify	Logical. If TRUE and supported (X11 or quartz devices), the plot will be interactive and clicking on a point will identify the sample the point represents. Press ESC or right-click on the plot screen to exit.
verbose	Logical. If TRUE, a message indicating where the pdf is saved will be printed on screen.

Value

The expression profiles plot, saved in the working directory as 'fileName.pdf'.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

Examples

```
######
# Load libraries and expression data
######

# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)

######
# Generic expression profile plot
######
# Plot expression of specific genes:
selectedGenes <- c("ENSG00000169575","ENSG00000078399","ENSG00000005381","ENSG00000154511")
plotExpressionProfiles(leukemiasEset, genes=selectedGenes, sampleLabels="LeukemiaType", type="boxplot")

# Color samples:
plotExpressionProfiles(leukemiasEset, genes="ENSG00000078399",
 sampleLabels="LeukemiaType",
 showMean=TRUE, identify=FALSE,
 sampleColors=c("grey","red")
 [(sampleNames(leukemiasEset) %in% c("GSM331386.CEL","GSM331392.CEL"))+1])

# Color classes:
plotExpressionProfiles(leukemiasEset, genes="ENSG00000078399",
 sampleLabels="LeukemiaType",
 showMean=TRUE, identify=TRUE,
 classColors=c("red", "blue", "red", "red","red"))

######
# Expression profiles related to a classifier
######
# Train a classifier or load a trained one:
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])
# leukemiasClassifier <- geNetClassifier(leukemiasEset[,trainSamples],
#    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier")
data(leukemiasClassifier) # Sample trained classifier

# Plot expression of the selected genes in the train samples:
plotExpressionProfiles(leukemiasEset[,trainSamples], leukemiasClassifier,
    sampleLabels="LeukemiaType", fileName="leukExprs.pdf")
```

```
# Plot expression of all the genes of specific classes:
classGenes <- getRanking(leukemiasClassifier@classificationGenes,
    showGeneID=TRUE)$geneID[,c("CLL"), drop=FALSE] # Feel free to modify
plotExpressionProfiles(leukemiasEset, genes=classGenes, sampleLabels="LeukemiaType",
    type="boxplot")

# Plot (on screen) the expression of the top ranked genes of each class
plotExpressionProfiles(leukemiasEset, leukemiasClassifier, sampleLabels="LeukemiaType")
```

`plotNetwork` *Plot GenesNetwork*

Description

Plots the coexpression and/or mutual information network for the given genes.

Usage

```
plotNetwork(genesNetwork, classificationGenes=NULL, genesRanking=NULL,
genesInfo=NULL, geneLabels=NULL, returniGraphs=FALSE,
plotType="dynamic", fileName=NULL,
plotAllNodesNetwork=TRUE, plotOnlyConnectedNodesNetwork=FALSE,
plotClassifcationGenesNetwork=FALSE,
labelSize=0.5, vertexSize=NULL, width=NULL, height=NULL, verbose=TRUE)
```

Arguments

`genesNetwork`	List of GenesNetwork returned by `geNetClassifier`. (`@genesNetwork`)
`classificationGenes`	Matrix or classificationGenes returned by `geNetClassifier`. (`@classificationGenes`)
`genesRanking`	Matrix or genesRanking returned by `geNetClassifier`. (`@genesRanking`)
`genesInfo`	List or data.frame with the properties of the genes to plot: `genesDetails(_@genesRanking)`
`geneLabels`	Vector or Matrix. Gene name, ID or label which should be shown in the returned results and plots.
`returniGraphs`	deprecated. A list with the plotted networks as igraph objects is always returned (see `invisible`), assign it to a variable if needed.
`plotType`	Character. "dynamic": Interactive plot. "static": One canvas split for the different networks. "pdf": All the networks are saved into a pdf file.
`fileName`	Character. File name to save the plot with. If not provided, the plots will be shown through the standard output device.
`plotAllNodesNetwork`	Logical. If TRUE, plots a network only with all the available genes
`plotOnlyConnectedNodesNetwork`	Logical. If TRUE, plots a network only with the connected nodes/genes

plotClassifcationGenesNetwork	Logical. If TRUE, plots a network only with the classification genes
labelSize	Integer. Gene/node label size for static and pdf plots.
vertexSize	Integer. Vertex minimum size.
width	Numeric. Dinamic or pdf plot width.
height	Numeric. Dinamic or pdf plot height.
verbose	Logical. If TRUE, messages indicating the execution progress will be shown.

Value

Graph list	List with the plotted igraph objects.
Network plots	Shown throught the standard output devide or saved in the working directory as 'fileName.pdf' if fileName was provided.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

References

Main package function and classifier training: geNetClassifier

Package igraph

See Also

plot.GenesNetwork() is an alias to this function. It can allso be called as i.e. plotNetwork(clGenesSubNet$ALL)
Note: The slot @genesNetwork returned by geNetClassifier is a List of GenesNetworks!

Examples

```
data(leukemiasClassifier)

# Step 1: Select a network or sub network
# Sub-network containing only the classification genes:
clGenesSubNet <- getSubNetwork(leukemiasClassifier@genesNetwork,
    leukemiasClassifier@classificationGenes)
# Step 2: Select the details/info about the genes to plot
# Classification genes' info:
clGenesInfo <- genesDetails(leukemiasClassifier@classificationGenes)

# Step 3: Plot the network
# Network plots can be interactive or plotted as PDF file.
#  - - Use plotType="pdf" to save the network as a static pdf file.
#       This option is recommended for getting an overview of several networks.
#  - - To get an interactive network, just skip this argument.

# Plot ALL network:
```

```
plotNetwork(clGenesSubNet$ALL, genesInfo=clGenesInfo)

# Plot AML network containing only the conected nodes:
plotNetwork(clGenesSubNet$ALL, genesInfo=clGenesInfo,
 plotAllNodesNetwork=FALSE, plotOnlyConnectedNodesNetwork=TRUE)

# The equivalent code to the plot geNetClassifier creates by default is:
topRanking <- getTopRanking(leukemiasClassifier@genesRanking, numGenesClass=100)
netTopGenes <- getSubNetwork(leukemiasClassifier@genesNetwork,
 getRanking(topRanking, showGeneID=TRUE)$geneID)
plotNetwork(netTopGenes,  classificationGenes=leukemiasClassifier@classificationGenes,
 genesRanking=topRanking, plotAllNodesNetwork=TRUE,
 plotOnlyConnectedNodesNetwork=TRUE, plotType="pdf",
 labelSize=0.3, fileName="leukemiasClassifier")

# In order to save the network as text file, you can use:
network2txt(leukemiasClassifier@genesNetwork, filePrefix="leukemiasNetwork")
```

`queryGeNetClassifier` *Queries the classifier trained with geNetClassifier.*

Description

Queries the classifier trained by geNetClassifier in order to find out the class of new samples.

Usage

```
queryGeNetClassifier(classifier, eset, minProbAssignCoeff = 1,
    minDiffAssignCoeff = 0.8, verbose = TRUE)
```

Arguments

`classifier`	Classifier returned by geNetClassifier. (@classifier)
`eset`	ExpressionSet or Matrix. Gene expression matrix of the new samples.
`minProbAssignCoeff`	Numeric. Coefficient to modify the minimum probability requird to assign a sample to a class. Reduce to improve call rate. Increase to reduce error. 0: Removes this restriction. The sample will always be assigned to the class with the highest probability. between 0 and 1: Reduces the required probability to assign a sample to a class. >1: Increases the required probability. Warning: if `minProbCoef` is equal to `2*number of classes`, all the samples will be left as 'NotAssigned'.
`minDiffAssignCoeff`	Numeric. Coefficient to modify the required difference between the two most likelly classes. Reduce to improve call rate. Increase to reduce error. 0: Removes this restriction. The probability of the second most-likely class will not be taken into account. between 1 and 1: Reduces the required difference to assign

the sample. >1: Increases the required difference. Warning: if `minDiffAssignCoeff` is equal to the number of classes, all the samples will be left as 'NotAssigned'.

`verbose` Logical. If TRUE, messages indicating the execution progress will be printed on screen.

Details

By default, in order to assign a sample two contitions must be met:

- if `minProbAssignCoeff` = `1`The probability of belonging to the class should be at least double of the random probability.
- if `minDiffAssignCoeff` = `0.8`The difference of probabilities between the most likely class and the second most likely class should be more than 80

  This means, that in a 4-class classifier, in order to assing a sample, the highest probabiity should be at least 0.5 (2x0.25), and the next most-likely-class should have a probability at least 0.2 (80 If these conditions are not met, the sample will be left as `notAssigned`.

  Modify the arguments values in order to modify these assignment conditions. Setting `minProbAssignCoeff` = `0` and `minDiffAssignCoeff` = `0` all samples will be assigned to the most likely class without any further restrictions.

Value

List:

- call Command used to execute the function.
- classes Classes to wich each of the samples were asigned to.
- probabilities Probabilities to the 2 classes each sample is most likelly to belong to.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

See Also

Main package function and classifier training: `geNetClassifier`
Query summary: `querySummary`
External validation stats: `externalValidation.stats` and `externalValidation.probMatrix`

Examples

```
#########################
## Classifier training
#########################

# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)
```

```
# Select the train samples:
# There should be the same number of samples from each class.
trainSamples<- c(1:10, 13:22, 25:34, 37:46, 49:58)
# summary(leukemiasEset$LeukemiaType[trainSamples])

# Train a classifier or load a trained one:
# leukemiasClassifier <- geNetClassifier(leukemiasEset[,trainSamples],
#    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier")
data(leukemiasClassifier) # Sample trained classifier

##########################
## Classifier Query
##########################
# Select the samples to query the classifier
#   - Real use: samples whose class we want to known
querySamples <- "GSM330154.CEL"
#   - External validation: samples not used for training
querySamples <- c(1:60)[-trainSamples]

#### Make a query to the classifier ("ask" about what class the new samples are):
queryResult <- queryGeNetClassifier(leukemiasClassifier, leukemiasEset[,querySamples])

# See the class it assigned to each sample:
queryResult$class[1:5]
# Or the samples which it wasn't sure about:
t(queryResult$probabilities[,queryResult$class=="NotAssigned"])

# Obtain an overview of the results
querySummary(queryResult)

#### Optional: Modify assignment conditions
# (minDiffCoef=0, minProbCoef=0: All samples will be assigned to the most likely class)
queryResult_AssignAll <- queryGeNetClassifier(leukemiasClassifier,
    leukemiasEset[,querySamples], minDiffAssignCoeff=0, minProbAssignCoeff=0)
# No samples are left as "NotAssigned":
queryResult$probabilities[,queryResult_AssignAll$class=="NotAssigned"]

#### External validation:
# Confusion matrix:
confMatrix <- table(leukemiasEset[,querySamples]$LeukemiaType,
    queryResult_AssignAll$class)
# New accuracy, call rate, sensitivity and specificity:
externalValidation.stats(confMatrix)
# Probability matrix for the assigned samples
externalValidation.probMatrix(queryResult, leukemiasEset[,querySamples]$LeukemiaType)
```

---

querySummary *Summary of the query.*

Description

Counts the number of samples assigned to each class and calculates basic statistics regarding the assignment probabilities.

Usage

```
querySummary(queryResult, showNotAssignedSamples = TRUE, numDecimals = 2,
    verbose = TRUE)
```

Arguments

queryResult	Object returned by queryGeNetClassifier
showNotAssignedSamples	Logical. Shows the two most likely classes for the NotAssigned samples and the probabilities of belonging to each of them.
numDecimals	Integer. Number of decimals to show on the statistics.
verbose	Logical. If TRUE, messages indicating the execution progress will be printed on screen.

Value

Returns a list with the following fields:

- callRate Count and percentage of assigned samples.
- assigned Number of samples assigned to each class and mean and SD of the assignment probabilities.
- notAssignedSamples Optional. Most likely classes for the Not Assigned samples.

Author(s)

Bioinformatics and Functional Genomics Group. Centro de Investigacion del Cancer (CIC-IBMCC, USAL-CSIC). Salamanca. Spain

See Also

Main package function and classifier training: geNetClassifier
Query the classifier: queryGeNetClassifier

Examples

```
#########################
## Classifier training
#########################

# Load an expressionSet:
library(leukemiasEset)
data(leukemiasEset)

# Select the train samples:
```

```
trainSamples <- c(1:10, 13:22, 25:34, 37:46, 49:58)

# Train a classifier or load a trained one:
# leukemiasClassifier <- geNetClassifier(leukemiasEset[,trainSamples],
#    sampleLabels="LeukemiaType", plotsName="leukemiasClassifier")
data(leukemiasClassifier) # Sample trained classifier

##########################
## Classifier query
##########################
# Select the samples to query the classifier
#   - Real use: samples whose class we want to known
querySamples <- "GSM330154.CEL"
#   - External validation: samples not used for training
querySamples <- c(1:60)[-trainSamples]

# Make a query to the classifier:
queryResult <- queryGeNetClassifier(leukemiasClassifier, leukemiasEset[,querySamples])


##########################
## Query Summary
##########################
# Obtain an overview of the results
querySummary(queryResult)
```

`setProperties-methods` *Set properties*

Description

Allows setting or modifiying the GenesRanking properties.

Methods

```
setProperties(object, geneLabels=NULL, discriminantPower=NULL,
meanDif=NULL, isRedundant=NULL, gERankMean=NULL)
```

See Also

Main package function and classifier training: `geNetClassifier`
This method's class (`GenesRanking`) help page.

Index

∗ **classes**
GeneralizationError-class, 10
GenesNetwork-class, 13
GenesRanking-class, 14
GeNetClassifierReturn-class, 22
∗ **classif**
calculateGenesRanking, 4
externalValidation.probMatrix, 7
externalValidation.stats, 8
geNetClassifier, 18
geNetClassifier-package, 2
leukemiasClassifier, 29
plot.GenesRanking, 33
plot.GeNetClassifierReturn, 34
plotAssignments, 35
plotDiscriminantPower, 37
plotExpressionProfiles, 39
plotNetwork, 41
queryGeNetClassifier, 43
querySummary, 45
∗ **leukemia**
leukemiasClassifier, 29
∗ **methods**
gClasses-methods, 10
genesDetails-methods, 12
getEdges-methods, 25
getNodes-methods, 25
getNumEdges-methods, 26
getNumNodes-methods, 26
getRanking-methods, 27
getSubNetwork-methods, 27
getTopRanking-methods, 28
network2txt, 30
numGenes-methods, 30
numSignificantGenes-methods, 31
overview-methods, 32
setProperties-methods, 47
∗ **package**
geNetClassifier-package, 2

calculateGenesRanking, *3*, 4, *15*
Classification genes' Discriminant Power, *35*

edges,GenesNetwork-method *(*getEdges-methods*)*, 25
externalValidation.probMatrix, *3*, 7, *9*, *44*
externalValidation.stats, *3*, *7*, 8, *44*
extractGenes,GenesRanking-method *(*GenesRanking-class*)*, 14

gClasses *(*gClasses-methods*)*, 10
gClasses,GenesRanking-method *(*gClasses-methods*)*, 10
gClasses-methods, 10
GeneralizationError, *4*, *23*, *32*
GeneralizationError *(*GeneralizationError-class*)*, 10
GeneralizationError-class, 10
genesDetails *(*genesDetails-methods*)*, 12
genesDetails,GenesRanking-method *(*genesDetails-methods*)*, 12
genesDetails-methods, 12
genesDetails.GenesRanking *(*genesDetails-methods*)*, 12
GenesNetwork, *3*, *23*, *25*, *26*, *28*, *30*, *32*
GenesNetwork *(*GenesNetwork-class*)*, 13
GenesNetwork-class, 13
GenesRanking, *3*, *5*, *10*, *13*, *23*, *27*, *28*, *31*, *32*, *47*
GenesRanking *(*GenesRanking-class*)*, 14
GenesRanking-class, 14
geneSymbols, 17
geNetClassifier, *3*, *7*, *9–11*, *13*, *15*, 18, *22*, *23*, *25–28*, *30–32*, *34–38*, *41*, *42*, *44*, *46*, *47*
geNetClassifier(), *23*
geNetClassifier-package, 2
GeNetClassifierReturn, *3*, *20*, *29*, *32*, *35*

GeNetClassifierReturn (GeNetClassifierReturn-class), 22
GeNetClassifierReturn-class, 22
getEdges (getEdges-methods), 25
getEdges,GenesNetwork-method (getEdges-methods), 25
getEdges-methods, 25
getNodes (getNodes-methods), 25
getNodes,GenesNetwork-method (getNodes-methods), 25
getNodes-methods, 25
getNumEdges (getNumEdges-methods), 26
getNumEdges,GenesNetwork-method (getNumEdges-methods), 26
getNumEdges-methods, 26
getNumNodes (getNumNodes-methods), 26
getNumNodes,GenesNetwork-method (getNumNodes-methods), 26
getNumNodes-methods, 26
getRanking (getRanking-methods), 27
getRanking,GenesRanking-method (getRanking-methods), 27
getRanking-methods, 27
getRanking.GenesRanking (getRanking-methods), 27
getSlots,GeNetClassifierReturn-method (GeNetClassifierReturn-class), 22
getSubNetwork (getSubNetwork-methods), 27
getSubNetwork,GenesNetwork-method (getSubNetwork-methods), 27
getSubNetwork,GeNetClassifierReturn-method (getSubNetwork-methods), 27
getSubNetwork,list-method (getSubNetwork-methods), 27
getSubNetwork,NULL-method (getSubNetwork-methods), 27
getSubNetwork-methods, 27
getTopRanking (getTopRanking-methods), 28
getTopRanking,GenesRanking-method (getTopRanking-methods), 28
getTopRanking-methods, 28
getTopRanking.GenesRanking (getTopRanking-methods), 28

initialize,GeneralizationError-method (GeneralizationError-class), 10
initialize,GenesNetwork-method (GenesNetwork-class), 13
initialize,GenesRanking-method (GenesRanking-class), 14
initialize,GeNetClassifierReturn-method (GeNetClassifierReturn-class), 22
invisible, *41*

leukemiasClassifier, 29
leukemiasEset, *4*

minet, *20*

names,GeNetClassifierReturn-method (GeNetClassifierReturn-class), 22
network2txt, 30
network2txt,GenesNetwork-method (network2txt), 30
network2txt,list-method (network2txt), 30
network2txt-methods (network2txt), 30
nodes,GenesNetwork-method (getNodes-methods), 25
numEdges,GenesNetwork-method (getNumEdges-methods), 26
numGenes (numGenes-methods), 30
numGenes,GenesRanking-method (numGenes-methods), 30
numGenes-methods, 30
numNodes,GenesNetwork-method (getNumNodes-methods), 26
numSignificantGenes (numSignificantGenes-methods), 31
numSignificantGenes,GenesRanking-method (numSignificantGenes-methods), 31
numSignificantGenes-methods, 31
numSignificantGenes.GenesRanking (numSignificantGenes-methods), 31

overview (overview-methods), 32
overview,GeneralizationError-method (overview-methods), 32
overview,GenesNetwork-method (overview-methods), 32

overview,GenesRanking-method
(overview-methods), 32
overview,GeNetClassifierReturn-method
(overview-methods), 32
overview-methods, 32

plot,GenesNetwork-method (plotNetwork),
41
plot,GenesRanking-method
(plot.GenesRanking), 33
plot,GeNetClassifierReturn-method
(plot.GeNetClassifierReturn),
34
plot.GenesNetwork (plotNetwork), 41
plot.GenesRanking, *5*, *15*, 33, *35*
plot.GeNetClassifierReturn, *23*, 34
plotAssignments, *3*, 35
plotDiscriminantPower, *3*, *35*, 37
plotExpressionProfiles, *3*, 39
plotGenesRanking (plot.GenesRanking), 33
plotGeNetClassifierReturn
(plot.GeNetClassifierReturn),
34
plotGeNetClassifierReturn,GeNetClassifierReturn-method
(plot.GeNetClassifierReturn),
34
plotNetwork, *3*, *13*, *35*, 41
plotNetwork,GenesNetwork-method
(plotNetwork), 41

queryGeNetClassifier, *3*, *7*, *9*, *19*, *21*, *36*,
43, *46*
querySummary, *3*, *7*, *44*, 45

setProperties (setProperties-methods),
47
setProperties,GenesRanking-method
(setProperties-methods), 47
setProperties-methods, 47
setProperties.GenesRanking
(setProperties-methods), 47
show,GeneralizationError-method
(GeneralizationError-class), 10
show,GenesNetwork-method
(GenesNetwork-class), 13
show,GenesRanking-method
(GenesRanking-class), 14
show,GeNetClassifierReturn-method
(GeNetClassifierReturn-class),
22

Significant genes, *35*
svm, *20*

Top ranked genes network (for each
class), *35*