Package 'Rmagpie'

November 3, 2025

Version 1.67.0

Title MicroArray Gene-expression-based Program In Error rate estimation

Author Camille Maumet <Rmagpie@gmail.com>, with contributions from C. Ambroise
J. Zhu

Maintainer Camille Maumet <Rmagpie@gmail.com>

Depends R (>= 2.6.1), Biobase (>= 2.5.5)

Imports Biobase (>= 2.5.5), e1071, graphics, grDevices, kernlab, methods, pamr, stats, utils

Suggests xtable

Description Microarray Classification is designed for both biologists and statisticians. It offers the ability to train a classifier on a labelled microarray dataset and to then use that classifier to predict the class of new observations. A range of modern classifiers are available, including
support vector machines (SVMs), nearest shrunken centroids (NSCs)... Advanced methods are provided to estimate
the predictive error rate and to report the subset of genes which appear essential in discriminating between classes.

License GPL (>= 3)

LazyLoad yes

URL http://www.bioconductor.org/

biocViews Microarray, Classification

collate AllClasses.R AllGenerics.R AllInitialize.R classifyNewSamples.R convertDataFile.R assessment-accessors.R featureSelectionOptions-accessors.R finalClassifier-accessors.R geneSubsets-accessors.R initialize-methods.R loadData-methods.R non_linear_svm.R one_layer_ext_CV.R rateGenes.R result2LayerCV-accessors.R resultRepeated1LayerCV-accessors.R rfe_zhu.R runOneLayerExtCV-methods.R runTwoLayerExtCV-methods.R setValidity-methods.R show-methods.R thresholds-accessors.R tool_case.R two_layer_ext_CV.R typeValidity.R user_interface.R

git_url https://git.bioconductor.org/packages/Rmagpie

git_branch devel

git_last_commit 531a8ca

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-11-02

Contents

`assessment-class` *assessment: A central class to perform one and two layers of external cross-validation on microarray data*

Description

This class stores the information relevant to a microarray classification assessment: data set, classifier and options are set here and then one-layer and two-layer cross-validation can be applied.

Creating objects

`new("assessment", dataset noFolds1stLayer=10, noFolds2ndLayer=9, classifierName="svm", featureSelectionMethod="rfe", typeFoldCreation="original", svmKernel="linear", noOfRepeat=2, featureSelectionOptions)`

Creates an assessment to be performed on the data set `dataset` using the feature selection options defined by `featureSelectionMethod` on the feature selection method `featureSelectionMethod` and with the classifier `classifierName`. Once all the options have been selected one-layer and two-layers of cross-validation can be performed by calling `runOneLayerExtCv` and `runTwoLayerExtCv` respectively.

`new("assessment", dataset noFolds1stLayer=10, noFolds2ndLayer=9, classifierName="svm", featureSelectionMethod="rfe", typeFoldCreation="original", svmKernel="linear", noOfRepeat=2)`

If `featureSelectionOptions` is not precised in the arguments then the options for the feature selection method are determined according to the `dataset` and the `featureSelectionMethod`. If RFE is selected as feature selection method then an object of class geneSubsets is automatically created. It defines sizes of subsets og genes for 1 to the number of features in the `dataset` by power of 2. If the feature selection method is NSC then the thresholds are taken to be the default thresholds generated by the function `pamr.train` from package `pamr` applied on `dataset`.

Slots

`dataset`: Object of class `"dataset"`. Microarray data set to be used for cross-validation

`noFolds1stLayer`: `numeric`. Number of folds in the inner layee layer of cross-validation

`noFolds2ndLayer`: `numeric`. Number of folds in one-layer cross-validation and in the second layer of cross-validation

`classifierName`: `character`. Name of the classifier: 'svm' for Support Vector Machines or 'nsc' for Nearest Shrunken Centroid

`featureSelectionMethod`: Object of class `"character"` ~~

`typeFoldCreation`: `character`. Type of fold creation: 'original', 'simple' or 'naive'

`svmKernel`: Object of class `"character"` ~~

`noOfRepeats`: `numeric`. Number of repeats to be performed for each cross-validation.

`featureSelectionOptions`: Object of class `"featureSelectionOptions"`. Sizes of subsets to be tried in the RFE or thresholds to be tried with the NSC.

`resultRepeated1LayerCV`: Object of class `"resultRepeated1LayerCVOrNULL"` NULL is the external one layer CV has not been run yet, resultRepeated1LayerCV containing the results

`resultRepeated2LayerCV`: Object of class `"result2LayerCVorNULL"` NULL is the external one layer CV has not been run yet, result2LayerCV containing the results

`finalClassifier`: Object of class `"finalClassifierOrNULL"` NULL is the final classifier has not been determined yet, finalClassifier containing the final Classifier for each feature selection option.

Methods

`classifyNewSamples(assessment)` Classify new samples using the final classifier. See related documentation.

`findFinalClassifier(assessment)` Train the final classifier related to an assessment based on each feature selection option. See related documentation

`getClassifierName(assessment)`, `getClassifierName(assessment)<-` Retrieve and Modify the classifier name associated to the current assessment (slot classifierName)

`getDataset(assessment)`, `getDataset(assessment)<-` Retrieve and Modify the dataset associated to the current assessment (slot dataset), see related documentation for more details.

`getFeatureSelectionOptions(assessment)`, `getFeatureSelectionOptions(assessment)<-` Retrieve and Modify the options of feature selection associated to the current assessment (slot featureSelectionOptions)

`getFinalClassifier(assessment)` Retreive the final classifier associated with an exeperiment.

`getNoFolds1stLayer(assessment)`, `getNoFolds1stLayer(assessment)<-` Retrieve and Modify the number of folds in the inner layer of cross-validation (slot nbFolds1stLayer)

`getNoFolds2ndLayer(assessment)`, `getNoFolds2ndLayer(assessment)<-` Retrieve and Modify the number of folds in the outer layer of cross-validation (slot nbFolds1stLayer)

`getNoOfRepeats(assessment)`, `getNoOfRepeats(assessment)<-` Retrieve and Modify the number of repeats of each cross-validation (slot nbOfRepeat)

`getResult1LayerCV(assessment)` Retrieve the results of the one-layer cross validation (slot resultRepeated1LayerCV). An easier access to this data is available via the method `getResults`)

`getResult2LayerCV(assessment)` Retrieve the results of the two-layers cross validation (slot result2LayerCV). An easier access to this data is available via the method `getResults`

getResults User-friendly methods to retreive data in the results of one-layer and two-layers of cross-validation. See related documentation page.

`getSvmKernel(assessment)`, `getSvmKernel(assessment)<-` Retrieve and Modify the svm kernel used as a final classifier if svm is the concerned classifier and during the Recusrsive Feature Elimination (slot svmKernel)

`getTypeFoldCreation(assessment)`, `getTypeFoldCreation(assessment)<-` Retrieve and Modify the type of folds creation to use for each cross-validation (slot typeFoldCreation)

`runOneLayerExtCV` Run one-layer cross-validation, see related documantation for more details.

`runTwoLayerExtCV` Run two-layer cross-validation, see related documantation for more details.

Author(s)

Camille Maumet

See Also

`geneSubsets`, `getResults-methods`, `runOneLayerExtCV-methods`, `runTwoLayerExtCV-methods`

Examples

```
#dataPath <- file.path("C:", "Documents and Settings", "c.maumet", "My Documents", "Programmation", "data")
#myDataset <- new("dataset", dataId="vantVeer_70", dataPath=file.path(dataPath, "vantVeer_70"))
# myDataset<-loadData(myDataset)
```

```
data('vV70genesDataset')

# assessment with RFE and SVM
myExpe <- new("assessment", dataset=vV70genes,
                  noFolds1stLayer=10,
                  noFolds2ndLayer=9,
                  classifierName="svm",
                  typeFoldCreation="original",
                  svmKernel="linear",
                  noOfRepeat=2,
                featureSelectionOptions=new("geneSubsets", optionValues=c(1,2,3,4,5,6)))

# Another assessment where the subsets are computed automatically
anotherExpe <- new("assessment",    dataset=vV70genes,
                                   noFolds1stLayer=10,
                                   noFolds2ndLayer=9,
                                   classifierName="svm",
                                   typeFoldCreation="original",
                                   svmKernel="linear",
                                   noOfRepeat=2)
getFeatureSelectionOptions(anotherExpe, topic='maxSubsetSize')
getFeatureSelectionOptions(anotherExpe, topic='subsetsSizes')

# assessment with NSC
expeWithNSC <- new("assessment",dataset=vV70genes,
                                noFolds1stLayer=10,
                                noFolds2ndLayer=9,
                                classifierName="nsc",
                                featureSelectionMethod='nsc',
                                typeFoldCreation="original",
                                svmKernel="linear",
                                noOfRepeat=2)
getFeatureSelectionOptions(expeWithNSC, topic='thresholds')
```

`classifyNewSamples-methods`

classifyNewSamples Method to classify new samples for a given assessment

---

Description

This method classify one or several new samples provided in the file 'newSamplesFile' using the final classifier build by 'findFinalClassifier'.

Arguments

`object`	`object` of `class` `assessment`. Object assessment of interest
`newSamplesFile`	`character`. URL of the file containing the gene expressions of the samples to be classified. The first line of the file must corresponds to the sample names and the first column to the names of the genes.

optionValue numeric. Size of subset (for RFE-SVM) or threshold (for NSC) to be considered, the option value must be available in the slot featureSelectionOptions of the assessment. If not, the smallest value bigger than 'optionValue' is selected. If this argument is missing the best option value according to one-layer cross-validation is used.

Methods

object = "assessment" This method is only applicable on objects of class assessment.

Examples

```
data('vV70genesDataset')

expeOfInterest <- new("assessment", dataset=vV70genes,
                                    noFolds1stLayer=10,
                                    noFolds2ndLayer=9,
                                    classifierName="svm",
                                    typeFoldCreation="original",
                                    svmKernel="linear",
                                    noOfRepeat=2,
                        featureSelectionOptions=new("geneSubsets", optionValues=c(1,2,4,8,16,32,64,70)))

# Build the final classifier
expeOfInterest <- findFinalClassifier(expeOfInterest)

## Not run:
classifyNewSamples(expeOfInterest, "pathToFile/testSamples_geneExpr.txt", 4)

## End(Not run)
expeOfInterest <- runOneLayerExtCV(expeOfInterest)
## Not run:
classifyNewSamples(expeOfInterest, "pathToFile/testSamples_geneExpr.txt")

## End(Not run)
```

---

featureSelectionOptions-class

"featureSelectionOptions": A virtual class to store the options of a feature selection

---

Description

This virtual class has two descendants: geneSubsets and thresholds. As a virtual class, you can't create an object of class featureSelectionOptions.

Slots

optionValues: numeric (vector). Value of the possible options

noOfOptions: numeric. Total number of options

Methods

`getOptionValues(featureSelectionOptions)` Retreive the value of options (slot optionValues)

`getNoOfOptions(featureSelectionOptions)` Retreive the number of options (slot featureSelectionOptions)

Author(s)

Camille Maumet

See Also

`geneSubsets`, `thresholds`

`finalClassifier-class` *finalClassifier: A class to store the final classifier corresponding to an assessment*

Description

This class stores the properties of the final classifiers associated to a given assessment. A classifier is usually available for each option value defined in the slot featureSelectionOptions. This final classifier is obtained by running the feature selction method on the whole dataset to find the relevant genes and then train the classifier on the whole data considering only the relevant genes.

Creating objects

To generate the final classifier, call the method 'findFinalClassifier' on an object of class assessment (`findFinalClassifier-methods`).

Slots

`genesFromBestToWorst:` `character`. If the feature selection method is RFE: the genes ordered by the weights obtained with the smallest subset size during RFE. If the method of featuure selection is the Nearest Shrunken Centroid, this slot is empty.

`models:` `list` of `object` of `class` `svm`.If the feature selection method is RFE: svm models trained on the whole dataset for each size of subset (2 attributes: 'model', the classifier model and 'modelFeatures' the features selected for each subset). If the feature selection method is NSC: the object created by pamr.train on the whole dataset.

Methods

`getGenesFromBestToWorst(finalClassifier)` Retreive the genes ordered by their weights obtained with the smallest subset during RFE (slot genesFromBestToWorst)

`getModels(finalClassifier)` Retreive the svm models for each size of subset (slot models)

Author(s)

Camille Maumet

See Also

finalClassifier,assessment, getFinalClassifier-methods

Examples

```
#dataPath <- file.path("C:", "Documents and Settings", "c.maumet", "My Documents", "Programmation", "Sources", "SV
#aDataset <- new("dataset", dataId="vantVeer_70", dataPath=dataPath)
#aDataset <- loadData(aDataset)
data('vV70genesDataset')

mySubsets <- new("geneSubsets", optionValues=c(1,2,4,8,16,32,64,70))
expeOfInterest <- new("assessment", dataset=vV70genes,
                                noFolds1stLayer=10,
                                noFolds2ndLayer=9,
                                classifierName="svm",
                                typeFoldCreation="original",
                                svmKernel="linear",
                                noOfRepeat=2,
                                featureSelectionOptions=mySubsets)

expeOfInterest <- findFinalClassifier(expeOfInterest)

# Return the whole object of class finalClassifier
finalClassifier <- getFinalClassifier(expeOfInterest)

# Svm model corresponding to a subset of size 4 (3rd size of subset)
getModels(finalClassifier)[[3]]$model
# Relevant genes for a subset of size 4 (3rd size of subset)
getModels(finalClassifier)[[3]]$modelFeatures

# Genes ordered according to their weight after performing the RFE up to 1 gene
getGenesFromBestToWorst(finalClassifier)
```

---

findFinalClassifier-methods

findFinalClassifier Method to train and build the final classifier based on an assessment

---

Description

This method generates and stores the final classifier corresponding to an assessment. This classifier can then be used to classify new samples by calling `classifyNewSamples`. The final classifier is build according to the classifier selected for a given assessment, applied on the whole data considering only the genes selected by the feature selction method selected.

Value

The methods returns an object of class assessment which finalClassifier has been build.

Methods

object = "assessment" This method is only applicable on objects of class assessment.

See Also

finalClassifier, assessment

Examples

```
#dataPath <- file.path("C:", "Documents and Settings", "c.maumet", "My Documents", "Programmation", "Sources", "SV
#aDataset <- new("dataset", dataId="vantVeer_70", dataPath=dataPath)
#aDataset <- loadData(aDataset)
data('vV70genesDataset')

# With the RFE-SVM as feature selection method
expeOfInterest <- new("assessment", dataset=vV70genes,
                                    noFolds1stLayer=10,
                                    noFolds2ndLayer=9,
                                    classifierName="svm",
                                    typeFoldCreation="original",
                                    svmKernel="linear",
                                    noOfRepeat=2,
                        featureSelectionOptions=new("geneSubsets", optionValues=c(1,2,4,8,16,32,64,70)))

# Build the final classifier
expeOfInterest <- findFinalClassifier(expeOfInterest)

# With the NSC as feature selection method
expeOfInterest <- new("assessment", dataset=vV70genes,
                                    noFolds1stLayer=10,
                                    noFolds2ndLayer=9,
                                    featureSelectionMethod="nsc",
                                    classifierName="nsc",
                                    typeFoldCreation="original",
                                    svmKernel="linear",
                                    noOfRepeat=2,
                                    featureSelectionOptions=new("thresholds"))

# Build the final classifier
expeOfInterest <- findFinalClassifier(expeOfInterest)
```

---

geneSubsets-class *geneSubsets: A class to handle the sizes of gene susbets to be tested during forward gene selection*

Description

Forward gene selection is usually a computationally expensive task. To reduce the computation expense one may want to do not consider one gene at a time but chunks of genes. This class store the sizes of gene susbets to be tested during forward gene selection.

Creating objects

`new("geneSubsets", optionValues)`

Create a geneSubsets, the sizes of the different subsets are determined by `optionValues`. The size of the biggest subset `maxSubsetSize` and the number of subsets to be tried `noOfOptions` are automatically deducted. The speed is set to `high` is there are less models than the size of the biggest subset and 'slow' if not.

`new("geneSubsets", maxSubsetSize, speed="high")`

Create a geneSubsets, with a biggest subset of size `maxSubsetSize`. If the `speed` is `high` the sizes of the subsets are increased by a power of 2 from 1 to the biggest power of 2 smaller than `maxSubsetSize`. If the speed is `slow` the sizes of the subsets are increased by 1 from 1 to the `maxSubsetSize`.

Slots

`maxSubsetSize`**:** `numeric`. Size of the biggest subset

`optionValues`**:** `numeric` (vector). Sizes of the subsets in acsending order

`noOfOptions`**:** `numeric`. Total number of subsets to be tried during backward gene selection

`speed`**:** `character`. Speed of the backward feature selection. `high` if the number of models is smaller than the size of the biggest subset, `slow` if not.

Methods

`getMaxSubsetSize(geneSubsets)`, `getMaxSubsetSize(geneSubsets)<-` Retreive and modify the size of the biggest subset (slot maxSubsetSize)

`getOptionValues(geneSubsets)`, `getOptionValues(geneSubsets)<-` Retreive and modify the sizes of the subsets of features (slot optionValues)

`getNoOfOptions(geneSubsets)` Retreive the total number of subsets to be tried during backward gene selection (slot noModels)

`getSpeed(geneSubsets)`, `getSpeed(geneSubsets)<-` Retreive and modify the speed of the backward feature selection. (slot speed)

Author(s)

Camille Maumet

See Also

`thresholds`,`assessment`

Examples

```
geneSubset235 <- new("geneSubsets", optionValues=c(2,3,5))
geneSubset235
getSubsetsSizes(geneSubset235)
getSpeed(geneSubset235)
getMaxSubsetSize(geneSubset235)

geneSubsetMax60 <- new("geneSubsets", maxSubsetSize=60, speed="slow")
geneSubsetMax60

geneSubsetSlow <- new("geneSubsets", maxSubsetSize=70, speed="slow")
geneSubsetSlow

getMaxSubsetSize(geneSubsetMax60) <- 70
geneSubsetMax60

newSizes <- c(1,2,3,4,5)
getSubsetsSizes(geneSubsetMax60) <- newSizes
geneSubsetMax60

getSpeed(geneSubset235) <- 'slow'
geneSubset235
```

`getDataset-methods` *getDataset Method to access the attributes of a dataset from an assessment*

Description

This method provides an easy interface to access the attributes of a dataset directly from an object assessment. The argument `topic` specifies which part of the dataset is of interest.

Arguments

`object`	`Object` of `class` `assessment`. Object assessment of interest
`topic`	`character`. Optional argument that specifies which attribute of the dataset is requested, the possible values are `"dataId"` (slot dataId of the dataset), `"dataPath"` (slot dataPath of the dataset), `"geneExprFile"` (slot geneExprFile of the dataset), `"classesFile"` (slot classesFile of the dataset), `"eset"` (slot eset of the dataset) if the `"topic"` is missing then the whole dataset object is returned.

Value

The value returned by the method changes accordingly to the `"topic"` argument.

If `"topic"` is missing `object` of `class` `dataset` the dataset corresponding to the assessment of interest

If `"topic"` is `"dataId"` `object` of `class` `character` corresponding to the `dataId` of the dataset

If "topic" is "dataPath" object of class character corresponding to the dataPath of the dataset

If "topic" is "geneExprFile" object of class character corresponding to the geneExprFile of the dataset

If "topic" is "classesFile" object of class character corresponding to the classesFile of the dataset

If "topic" is "eset" object of class ExpressionSetOrNull corresponding to the eset of the dataset

Methods

object = "assessment" The method is only applicable on objects of class assessment.

Author(s)

Camille Maumet

See Also

assessment

Examples

```
#dataPath <- file.path("C:", "Documents and Settings", "c.maumet", "My Documents", "Programmation", "Sources", "SV
#aDataset <- new("dataset", dataId="vantVeer_70", dataPath=dataPath)
#aDataset <- loadData(aDataset)

data('vV70genesDataset')

expeOfInterest <- new("assessment", dataset=vV70genes,
                                    noFolds1stLayer=10,
                                    noFolds2ndLayer=9,
                                    classifierName="svm",
                                    typeFoldCreation="original",
                                    svmKernel="linear",
                                    noOfRepeat=2,
                        featureSelectionOptions=new("geneSubsets", optionValues=c(1,2,3,4,5,6)))

getDataset(expeOfInterest)
```

---

getFeatureSelectionOptions-methods

getFeatureSelectionOptions Method to access the attributes of a featureSelectionOptions from an assessment

---

Description

This method provides an easy interface to access the attributes of the object of class featureSelectionOptions related to a particular assessment, directly from this object assessment. The argument `topic` specifies which part of the featureSelectionOptions is of interest.

Arguments

`object`	`Object` of `class` `assessment`. Object assessment of interest
`topic`	`character`. Optional argument that specifies which attribute of the featureSelectionOptions is requested, the possible values are: `"optionValues"` (Access the slot `optionValues` of the featureSelectionOptions), `"noOfOptions"` (Access the slot `noOfOptions` of the featureSelectionOptions), if the featureSelectionOptions object is an object of class `geneSubsets`, then the following values are also available for the argument `topic` `"subsetsSizes"` (Access the slot `optionValues` of the geneSubsets), `"noModels"` (Access the slot `noOfOptions` of the geneSubsets), `"maxSubsetSize"` (Access the slot `maxSelectedFeatures` of the geneSubsets), `"speed"` (Access the slot `speed` of the featureSelectionOptions), if the featureSelectionOptions object is an object of class `thresholds`, then the following values are also available for the argument `topic` `"thresholds"` (Access the slot `optionValues` of the object thresholds), `"noThresholds"` (Access the slot `noOfOptions` of the object thresholds) if the `topic` is missing then the whole featureSelectionOptions object is returned.

Value

The value returned by the method changes accordingly to the 'topic' argument.

If `topic` is missing `object` of `class` `featureSelectionOptions` the featureSelectionOptions corresponding to the assessment of interest

If `topic` is "optionValues" `numeric` corresponding to the `optionValues` of the featureSelectionOptions

If `topic` is "noOfOptions" `numeric` corresponding to the `noOfOptions` of the featureSelectionOptions

If `object` is of class `geneSubsets` and `topic` is "maxSubsetSize" `numeric` corresponding to the `maxSubsetSize` of the geneSubsets

If `object` is of class `geneSubsets` and `topic` is "subsetsSizes" `numeric` corresponding to the `optionValues` of the geneSubsets

If `object` is of class `geneSubsets` and `topic` is "noModels" `numeric` corresponding to the `noOfOptions` of the geneSubsets

If `object` is of class `geneSubsets` and `topic` is "speed" `numeric` corresponding to the `speed` of the geneSubsets

If `object` is of class `thresholds` and `topic` is "thresholds" `numeric` corresponding to the `optionValues` of the object of class thresholds

If `object` is of class `thresholds` and `topic` is "noThresholds" `numeric` corresponding to the `noOfOptions` of the object of class thresholds

Methods

object = "assessment" The method is only applicable on objects of class assessment.

Author(s)

Camille Maumet

See Also

featureSelectionOptions, assessment

Examples

```
# With an assessment using RFE
#dataPath <- file.path("C:", "Documents and Settings", "c.maumet", "My Documents", "Programmation", "Sources", "SV
#aDataset <- new("dataset", dataId="vantVeer_70", dataPath=dataPath)
#aDataset <- loadData(aDataset)

data('vV70genesDataset')

mySubsets <- new("geneSubsets", optionValues=c(1,2,3,4,5,6))
myExpe <- new("assessment", dataset=vV70genes,
                                    noFolds1stLayer=10,
                                    noFolds2ndLayer=9,
                                    classifierName="svm",
                                    typeFoldCreation="original",
                                    svmKernel="linear",
                                    noOfRepeat=2,
                                    featureSelectionOptions=mySubsets)

# Return the whole object 'featureSelectionOptions' (an object of class geneSusbsets)
getFeatureSelectionOptions(myExpe)
# Size of the biggest subset
getFeatureSelectionOptions(myExpe, topic='maxSubsetSize')
# All sizes of subsets
getFeatureSelectionOptions(myExpe, topic='subsetsSizes')
# Speed
getFeatureSelectionOptions(myExpe, topic='speed')
# Number of subsets
getFeatureSelectionOptions(myExpe, topic='noModels') == getNoModels(mySubsets)

# With an assessment using NSC as a feature selection method
myThresholds <- new("thresholds", optionValues=c(0.1,0.2,0.3))
myExpe2 <- new("assessment", dataset=vV70genes,
                                    noFolds1stLayer=10,
                                    noFolds2ndLayer=9,
                                    classifierName="nsc",
                                    featureSelectionMethod='nsc',
                                    typeFoldCreation="original",
                                    svmKernel="linear",
                                    noOfRepeat=2,
                                    featureSelectionOptions=myThresholds)
```

```
# Return the whole object 'featureSelectionOptions' (an object of class geneSusbsets)
getFeatureSelectionOptions(myExpe2)
# vector of thresholds
getFeatureSelectionOptions(myExpe2, topic='thresholds')
# Number of thresholds
getFeatureSelectionOptions(myExpe2, topic='noThresholds')
```

`getFinalClassifier-methods`

getFinalClassifier Method to access the attributes of a finalClassifier from an assessment

---

Description

This method provides an easy interface to access the attributes of the object of class finalClassifier related to a particular assessment, directly from this object assessment. The argument `topic` specifies which part of the finalClassifier is of interest.

Arguments

`object`	`Object` of `class` `assessment`. Object assessment of interest
`topic`	`character`. Optional argument that specifies which attribute of the finalClassifier is requested, the possible values are `genesFromBestToWorst` (slot `genesFromBestToWorst` of the finalClassifier), `models` (slot `models` of the finalClassifier), if the `topic` is missing then the whole finalClassifier object is returned.

Value

The value returned by the method changes accordingly to the `topic` argument.

If `topic` is missing `object` of `class` `finalClassifier` the finalClassifier corresponding to the assessment of interest

If `topic` is `"genesFromBestToWorst"` `numeric` corresponding to the `genesFromBestToWorst` of the finalClassifier

If `topic` is `"models"` `numeric` corresponding to the `models` of the finalClassifier

Methods

object = "assessment" The method is only applicable on objects of class assessment.

Author(s)

Camille Maumet

See Also

`finalClassifier`, `assessment`

Examples

```
#dataPath <- file.path("C:", "Documents and Settings", "c.maumet", "My Documents", "Programmation", "Sources", "SV
#aDataset <- new("dataset", dataId="vantVeer_70", dataPath=dataPath)
#aDataset <- loadData(aDataset)

mySubsets <- new("geneSubsets", optionValues=c(1,2,3,4,5,6))
data('vV70genesDataset')

# assessment with RFE and SVM
expeOfInterest <- new("assessment", dataset=vV70genes,
                              noFolds1stLayer=10,
                              noFolds2ndLayer=9,
                              classifierName="svm",
                              typeFoldCreation="original",
                              svmKernel="linear",
                              noOfRepeat=2,
                              featureSelectionOptions=mySubsets)

expeOfInterest <- findFinalClassifier(expeOfInterest)

# Return the whole object of class finalClassifier
getFinalClassifier(expeOfInterest)
getFinalClassifier(expeOfInterest, 'genesFromBestToWorst')
getFinalClassifier(expeOfInterest, 'models')

# assessment with NSC
expeOfInterest <- new("assessment", dataset=vV70genes,
                              noFolds1stLayer=10,
                              noFolds2ndLayer=9,
                              featureSelectionMethod='nsc',
                              classifierName="nsc",
                              typeFoldCreation="original",
                              svmKernel="linear",
                              noOfRepeat=2,
                              featureSelectionOptions=new("thresholds"))

expeOfInterest <- findFinalClassifier(expeOfInterest)

# Return the whole object of class finalClassifier
getFinalClassifier(expeOfInterest)
getFinalClassifier(expeOfInterest, 'genesFromBestToWorst')
getFinalClassifier(expeOfInterest, 'models')
```

`getResults-methods` *getResults Method to access the result of one-layer and two-layers cross-validation from an assessment*

Description

This method provides an easy interface to access the results of one-layer and two-layers of cross-validation directly from an object assessment.

Arguments

object	Object of class assessment. Object assessment of interest
layer	numeric. Indice that states which layer of cross-validation must be accessed. Set to 1 to acces the one-layer cross-validation, Set to c(1,i) to acces the ith repeat of the one-layer cross-validation, Set to 2 to acces to the two-layers cross-validation, Set to c(2,i) to access the ith repeat of the two-layers cross-validation, Set to c(2,i,j) to access the jth inner layer of ith repeat of the two-layers cross-validation, Set to c(2,i,j,k) to access the kth repeat of the jth inner layer of ith repeat of the two-layers cross-validation
topic	character. Argument that specifies which kind of result is requested, the possible values are "errorRate": Access to cross-validation error rate, standard error on cross-validated error rate, error rate per fold, number of samples per fold and error rate per class, "selectedGenes": Access to the genes selected for each fold or their frequency of selection among the folds and the repeats, "bestOptionValue": For one-layer of cross-validation, access to the best option value (size of gene subset for SVM-RFE or thresholds for NSC) corresponding to the best value of the cross-validated error rate. For the two-layers of cross-validation, access the average best option value (over the repeats and folds). "executionTime": Time used to run the selected layer in seconds.
errorType	character. Optional, ignored if topic is not "errorRate". Specify the type of error rate requested, the possible values are: missing or "all" to access all the following error rates "cv" to access the cross-validated error rate, "se" to access the standard error on the cross validated error rate, "fold" to access the error rate per fold (not available in certain cases see section value for more details), "noSamplesPerFold" to access the number of samples in each fols (not available in certain cases see section value for more details), "class" to acces the error rate per class
genesType	character. Optional, ignored if topic is not "selectedGenes". Specify the type of display of genes selected, the possible values are: missing "fold" to access the genes selected for each fold (not available in certain case see section value for more details), "frequ" to access the genes order by their frequency among the folds(not available in certain case see section value for more details)

Value

if there is no error, the value returned by the method depends on the arguments namely, layer, topic, errorType and genesType.

If layer is 1

General	Get the results of the repeated one-layer cross-validation corresponding to the object of class assessment. If the one-layer cross-validation has not been performed and the user try to access it then the function return an error indicating that he must call runOneLayerExtCV first.

if topic is "errorRate"

If errorType="all" or is missing

All the following error rates

If errorType="cv"

numeric. Cross-validated error-rate for each value of option tried obtained by one-layer of cross-validation (1 value per value of option).

If errorType="se"

numeric. Standard error on cross-validated error-rate for each value of option tried obtained by one-layer of cross-validation (1 value per value of option).

If errorType="class"

numeric. Class cross-validated error rate error for each value of option tried obtained by one-layer of cross-validation (1 value per class and value of option).

Else Error signaling that the topic is not appropriate.

if topic is "genesSelected"

If genesType="freq" or is missing

list. Each elelement of the list corresponds to the genes selected for each model ordered by frequency.

Else Error signaling that the topic is not appropriate.

if topic is "bestOptionValue"

Size of subset (for RFE-SVM) or threshold (for NSC) corresponding to the minimum cross-validated error rate.

if topic is "executionTime"

Time in second to perform this one-layer cross-validation.

If layer is c(1,i)

General Get the results of the ith repeat of the one-layer cross-validation corresponding to the object of class assessment. If the one-layer cross-validation has not been performed and the user try to access it then the function return an error indicating that he must call runOneLayerExtCV first.

if topic is "errorRate"

If errorType="all" or is missing

All the following error rates

If errorType="cv"

numeric. Cross-validated error-rate for each value of option tried obtained by one-layer of cross-validation on the ith repeat(1 value per subset).

If errorType="se"

numeric. Standard error on cross-validated error-rate for each value of option tried obtained by one-layer of cross-validation on the ith repeat (1 value per value of option).

If errorType="class"

numeric. Class cross-validated error rate error for each value of option tried obtained by one-layer of cross-validation on the ith repeat (1 value per class and value of option).

If errorType="fold"

numeric. Class cross-validated error rate error for each fold and each value of option tried obtained by one-layer of cross-validation on the ith repeat (1 value per class and value of option).

`Else` Error signaling that the topic is not appropriate.

`if topic is "genesSelected"`

`If genesType="freq" or is missing`
: list. Each elelement of the list corresponds to the genes selected for each model ordered by frequency.

`If genesType="fold"`
: list. Each elelement of the list corresponds to a model and contains a list of which one element correspond to the genes selected in a particular fold.

`Else` Error signaling that the topic is not appropriate.

`if topic is "bestOptionValue"`
: numeric. Size of subset (for RFE) or threshold (for NSC) corresponding to the minimum cross-validated error rate in the ith repeat of the one-layer cross-validation.

`if topic is "executionTime"`
: Time in second to perform this repeat of one-layer cross-validation.

If `layer` is 2

`General` Get the results of the repeated two-layers cross-validation corresponding to the `object` of class assessment. If the two-layer cross-validation has not been performed and the user try to access it then the function return an error indicating that he must call `runTwoLayerExtCV` first.

`if topic is 'errorRate'`

`If errorType="all" or is missing`
: All the following error rates

`If errorType="cv"`
: numeric. Cross-validated error-rate obtained by two-layers of cross-validation (1 value).

`If errorType="se"`
: numeric. Standard error on cross-validated error-rate obtained by two-layers of cross-validation (1 value).

`If errorType="class"`
: numeric. Class cross-validated error rate obtained by two-layers (1 value per class)

`Else` Error signaling that the topic is not appropriate.

`if topic is "bestOptionValue"`
: numeric. Average best number of genes for SVM-RFE of threshold for NSc obtained among the folds.

`if topic is "executionTime"`
: Time in second to perform this two-layers cross-validation.

If `layer` is c(2,i)

`General` Get the results of the ith repeated of the two-layers cross-validation corresponding to the `object` of class assessment. If the two-layer cross-validation has not been performed and the user try to access it then the function return an error indicating that he must call `runTwoLayerExtCV` first.

`if topic is 'errorRate'`

`If errorType="all" or is missing`
: All the following error rates

`If errorType="cv"`
: numeric. Cross-validated error-rate obtained by two-layers of cross-validation in this repeat. (1 value).

`If errorType="se"`
: numeric. Standard error on cross-validated error-rate obtained by two-layers of cross-validation in this repeat (1 value).

`If errorType="class"`
: numeric. Class cross-validated error rate obtained by two-layers in this repeat

`If errorType="fold"`
: numeric. Error rate obtained on each of the folds in the second layer in this repeat(1 value per fold). of cross-validation (value per class).

`Else`
: Error signaling that the topic is not appropriate.

`if topic is "genesSelected"`

`If genesType="fold" or is missing`
: list. Each elelement of the list corresponds to a fold and contains a list of the genes selected in this particular fold.

`Else`
: Error signaling that the topic is not appropriate.

`if topic is "bestOptionValue"`
: numeric. Average best number of genes obtained among the folds in this repeat.

`if topic is "executionTime"`
: Time in second to perform this repeat of two-layers cross-validation.

`If layer is c(2, i, j)`
: This layer corresponds to the jth inner layer of one-layer cross-validation performed inside the ith repeat of the two-layers cross-validation. The returned values are similar to the one returned by a repeated one-layer cross-validation.

`If layer is c(2, i, j, k)`
: This layer corresponds to the kth repeat of the jth inner layer of one-layer cross-validation performed inside the ith repeat. The returned values are similar to the one returned by a repeat of one-layer cross-validation.

Methods

object = "assessment" The method is only applicable on objects of class assessment.

Author(s)

Camille Maumet

See Also

`assessment`

Examples

```
#dataPath <- file.path("C:", "Documents and Settings", "c.maumet", "My Documents", "Programmation", "Sources", "SV
#aDataset <- new("dataset", dataId="vantVeer_70", dataPath=dataPath)
#aDataset <- loadData(aDataset)
data('vV70genesDataset')

mySubsets <- new("geneSubsets", optionValues=c(1,2,4,8,16,32,64,70))
myassessment <- new("assessment", dataset=vV70genes,
                                  noFolds1stLayer=5,
                                  noFolds2ndLayer=4,
                                  classifierName="svm",
                                  typeFoldCreation="original",
                                  svmKernel="linear",
                                  noOfRepeat=2,
                                  featureSelectionOptions=mySubsets)

myassessment <- runOneLayerExtCV(myassessment)
myassessment <- runTwoLayerExtCV(myassessment)

# --- Access to one-layer CV ---
# errorRate
# 1-layer CV: error Rates
getResults(myassessment, 1, 'errorRate')
# 1-layer CV: error Rates - all")
getResults(myassessment, 1, 'errorRate', errorType='all')
# 1-layer CV: error Rates - cv
getResults(myassessment, 1, 'errorRate', errorType='cv')
# 1-layer CV: error Rates - se
getResults(myassessment, 1, 'errorRate', errorType='se')
# 1-layer CV: error Rates - class
getResults(myassessment, 1, 'errorRate', errorType='class')

# genesSelected
# 1-layer CV: genes Selected
getResults(myassessment, 1, 'genesSelected')
# 1-layer CV: genes Selected - frequ
getResults(myassessment, 1, 'genesSelected', genesType='frequ')
# 1-layer CV: genes Selected - model 7
getResults(myassessment, 1, 'genesSelected', genesType='frequ')[[7]]
getResults(myassessment, 1, 'genesSelected')[[7]]

# bestOptionValue
# 1-layer CV: best number of genes
getResults(myassessment, 1, 'bestOptionValue')

# executionTime
# 1-layer CV: execution time
getResults(myassessment, 1, 'executionTime')

# --- Access to 2nd repeat of one-layer CV ---
# Error rates
# 1-layer CV repeat 2: error Rates
```

```
getResults(myassessment, c(1,2), 'errorRate')
# 1-layer CV repeat 2: error Rates - all
getResults(myassessment, c(1,2), 'errorRate', errorType='all')
# 1-layer CV repeat 2: error Rates - cv
getResults(myassessment, c(1,2), 'errorRate', errorType='cv')
# 1-layer CV repeat 2: error Rates - se
getResults(myassessment, c(1,2), 'errorRate', errorType='se')
# 1-layer CV repeat 2: error Rates - fold
getResults(myassessment, c(1,2), 'errorRate', errorType='fold')
# 1-layer CV repeat 2: error Rates - noSamplesPerFold
getResults(myassessment, c(1,2), 'errorRate', errorType='noSamplesPerFold')
# 1-layer CV repeat 2: error Rates - class
getResults(myassessment, c(1,2), 'errorRate', errorType='class')

# genesSelected
# 1-layer CV repeat 2: genes Selected
getResults(myassessment, c(1,2), 'genesSelected')
# 1-layer CV repeat 2: genes Selected - frequ
getResults(myassessment, c(1,2), 'genesSelected', genesType='frequ')
# 1-layer CV repeat 2: genes Selected - model 7 (twice)
getResults(myassessment, c(1,2), 'genesSelected', genesType='frequ')[[7]]
getResults(myassessment, c(1,2), 'genesSelected')[[7]]
# 1-layer CV repeat 2: genes Selected - fold
getResults(myassessment, c(1,2), 'genesSelected', genesType='fold')

# 1-layer CV repeat 2: best number of genes
getResults(myassessment, c(1,2), 'bestOptionValue')

# 1-layer CV repeat 2: execution time
getResults(myassessment, c(1,2), 'executionTime')

# --- Access to two-layers CV ---
# Error rates
# 2-layer CV: error Rates
getResults(myassessment, 2, 'errorRate')
# 2-layer CV: error Rates - all
getResults(myassessment, 2, 'errorRate', errorType='all')
# 2-layer CV: error Rates - cv
getResults(myassessment, 2, 'errorRate', errorType='cv')
# 2-layer CV: error Rates - se
getResults(myassessment, 2, 'errorRate', errorType='se')
# 2-layer CV: error Rates - class
getResults(myassessment, 2, 'errorRate', errorType='class')

# bestOptionValue
# 2-layer CV: best number of genes (avg)
getResults(myassessment, 2, 'bestOptionValue')

# executionTime
# 2-layer CV: execution time
getResults(myassessment, 2, 'executionTime')

# --- Access to two-layers CV access to repeats ---
```

```
# Error rates
# 2-layer CV repeat 1: error Rates
getResults(myassessment, c(2,1), 'errorRate')
# 2-layer CV repeat 1: error Rates - all
getResults(myassessment, c(2,1), 'errorRate', errorType='all')
# 2-layer CV repeat 1: error Rates - cv
getResults(myassessment, c(2,1), 'errorRate', errorType='cv')
# 2-layer CV repeat 1: error Rates - se
getResults(myassessment, c(2,1), 'errorRate', errorType='se')
# 2-layer CV repeat 1: error Rates - fold
getResults(myassessment, c(2,1), 'errorRate', errorType='fold')
# 2-layer CV repeat 1: error Rates - noSamplesPerFold
getResults(myassessment, c(2,1), 'errorRate', errorType='noSamplesPerFold')
# 2-layer CV repeat 1: error Rates - class
getResults(myassessment, c(2,1), 'errorRate', errorType='class')

# genesSelected
# 2-layer CV repeat 1: genes Selected
getResults(myassessment, c(2,1), 'genesSelected')
# 2-layer CV repeat 1: genes Selected - fold
getResults(myassessment, c(2,1), 'genesSelected', genesType='fold')

# 2-layer CV repeat 1: best number of genes
getResults(myassessment, c(2,1), 'bestOptionValue')

# 2-layer CV repeat 1: execution time
getResults(myassessment, c(2,1), 'executionTime')

# --- Access to one-layer CV inside two-layers CV ---
# errorRate
# 2-layer CV repeat 1 inner layer 3: error Rates
getResults(myassessment, c(2,1,3), 'errorRate')
# 2-layer CV repeat 1 inner layer 3: error Rates - all
getResults(myassessment, c(2,1,3), 'errorRate', errorType='all')
# 2-layer CV repeat 1 inner layer 3: error Rates - cv
getResults(myassessment, c(2,1,3), 'errorRate', errorType='cv')
# 2-layer CV repeat 1 inner layer 3: error Rates - se
getResults(myassessment, c(2,1,3), 'errorRate', errorType='se')
# 2-layer CV repeat 1 inner layer 3: error Rates - class
getResults(myassessment, c(2,1,3), 'errorRate', errorType='class')

# genesSelected
# 2-layer CV repeat 1 inner layer 3: genes Selected
getResults(myassessment, c(2,1,3), 'genesSelected')
# 2-layer CV repeat 1 inner layer 3: genes Selected - frequ
getResults(myassessment, c(2,1,3), 'genesSelected', genesType='frequ')
# 2-layer CV repeat 1 inner layer 3: genes Selected - model 7
getResults(myassessment, c(2,1,3), 'genesSelected', genesType='frequ')[[7]]
getResults(myassessment, c(2,1,3), 'genesSelected')[[7]]

# bestOptionValue
# 2-layer CV repeat 1 inner layer 3: best number of genes
getResults(myassessment, c(2,1,3), 'bestOptionValue')
```

```
# executionTime
# 2-layer CV repeat 1 inner layer 3: execution time
getResults(myassessment, c(2,1,3), 'executionTime')

 # --- two-layers CV access to repeat 1, inner layer 2 repeat 2 ---
# Error rates
# 2-layer CV inner layer 3 repeat 2: error Rates
getResults(myassessment, c(2,1,3,1), 'errorRate')
# 2-layer CV repeat 1 inner layer 3 repeat 1: error Rates - all
getResults(myassessment, c(2,1,3,1), 'errorRate', errorType='all')
# 2-layer CV repeat 1 inner layer 3 repeat 1: error Rates - cv
getResults(myassessment, c(2,1,3,1), 'errorRate', errorType='cv')
# 2-layer CV repeat 1 inner layer 3 repeat 1: error Rates - se
getResults(myassessment, c(2,1,3,1), 'errorRate', errorType='se')
# 2-layer CV repeat 1 inner layer 3 repeat 1: error Rates - class
getResults(myassessment, c(2,1,3,1), 'errorRate', errorType='class')
# 2-layer CV repeat 1 inner layer 3 repeat 1: error Rates - fold
getResults(myassessment, c(2,1,3,1), 'errorRate', errorType='fold')
# 2-layer CV repeat 1 inner layer 3 repeat 1: error Rates - noSamplesPerFold
getResults(myassessment, c(2,1,3,1), 'errorRate', errorType='noSamplesPerFold')

# genesSelected
# 2-layer CV repeat 1 inner layer 3 repeat 1: genes Selected
getResults(myassessment, c(2,1,3,1), 'genesSelected')
# 2-layer CV repeat 1 inner layer 3 repeat 1: genes Selected - fold
getResults(myassessment, c(2,1,3,1), 'genesSelected', genesType='fold')
# 2-layer CV repeat 1 inner layer 3 repeat 1: genes Selected - model 3 fold 1(twice)
getResults(myassessment, c(2,1,3,1), 'genesSelected', genesType='fold')[[3]][[1]]
# 2-layer CV repeat 1 inner layer 3 repeat 1: genes Selected frequ - model 3
getResults(myassessment, c(2,1,3,1), 'genesSelected')[[3]]

# 2-layer CV repeat 1 inner layer 3 repeat 1: best number of genes
getResults(myassessment,  c(2,1,3,1), 'bestOptionValue')

# 2-layer CV repeat 1 inner layer 3 repeat 1: execution time
getResults(myassessment,  c(2,1,3,1), 'executionTime')
```

`initialize-methods` *Initialize objects of class from Rmagpie*

Description

Initialize method for Rmagpie classes.

Author(s)

Camille Maumet

plotErrorsFoldTwoLayerCV-methods
plotErrorsFoldTwoLayerCV Method to plot the error rate of a two-layer Cross-validation

Description

This method creates a plot that reprenset the error rate in each fold of each repeat of the second layer of cross-validation of the two-layer cross-validation of the assessment at stake. The plot represents the error rate versus the size of gene subsets (for SVM-RFE) or the threshold values (for NSC).

Methods

object = "assessment" The method is only applicable on objects of class assessment.

See Also

plotErrorsSummaryOneLayerCV-methods, plotErrorsRepeatedOneLayerCV-methods

Examples

```
data('vV70genesDataset')

expeOfInterest <- new("assessment", dataset=vV70genes,
                                    noFolds1stLayer=3,
                                    noFolds2ndLayer=2,
                                    classifierName="svm",
                                    typeFoldCreation="original",
                                    svmKernel="linear",
                                    noOfRepeat=10,
                        featureSelectionOptions=new("geneSubsets", optionValues=c(1,2,3,4,5,6)))

expeOfInterest <- runTwoLayerExtCV(expeOfInterest)

plotErrorsFoldTwoLayerCV(expeOfInterest)
```

plotErrorsRepeatedOneLayerCV-methods
plotErrorsRepeatedOneLayerCV Method to plot the estimated error rates in each repeat of a one-layer Cross-validation

Description

This method creates a plot that represent the summary estimated error rate and the cross-validated error rate in each repeat of the one-layer cross-validation of the assessment at stake. The plot represents the summary estimate of the error rate (averaged over the repeats) and the cross-validated error rate obtained in each repeat versus the size of gene subsets (for SVM-RFE) or the threshold values (for NSC).

Methods

object = "assessment" The method is only applicable on objects of class assessment.

See Also

plotErrorsFoldTwoLayerCV-methods, plotErrorsSummaryOneLayerCV-methods

Examples

```
data('vV70genesDataset')

expeOfInterest <- new("assessment", dataset=vV70genes,
                                    noFolds1stLayer=3,
                                    noFolds2ndLayer=2,
                                    classifierName="svm",
                                    typeFoldCreation="original",
                                    svmKernel="linear",
                                    noOfRepeat=10,
                          featureSelectionOptions=new("geneSubsets", optionValues=c(1,2,3,4,5,6)))

expeOfInterest <- runOneLayerExtCV(expeOfInterest)

plotErrorsRepeatedOneLayerCV(expeOfInterest)
```

---

plotErrorsSummaryOneLayerCV-methods

plotErrorsSummaryOneLayerCV Method to plot the summary estimated error rates of a one-layer Cross-validation

---

Description

This method creates a plot that represent the summary estimated error rate of the one-layer cross-validation of the assessment at stake. The plot represents the summary estimate of the error rate (averaged over the repeats) versus the size of gene subsets (for SVM-RFE) or the threshold values (for NSC).

Methods

object = "assessment" The method is only applicable on objects of class assessment.

See Also

plotErrorsFoldTwoLayerCV-methods, plotErrorsRepeatedOneLayerCV-methods

Examples

```
data('vV70genesDataset')

expeOfInterest <- new("assessment", dataset=vV70genes,
                                    noFolds1stLayer=3,
                                    noFolds2ndLayer=2,
                                    classifierName="svm",
                                    typeFoldCreation="original",
                                    svmKernel="linear",
                                    noOfRepeat=10,
                        featureSelectionOptions=new("geneSubsets", optionValues=c(1,2,3,4,5,6)))

expeOfInterest <- runOneLayerExtCV(expeOfInterest)

plotErrorsSummaryOneLayerCV(expeOfInterest)
```

---

rankedGenesImg-methods

rankedGenesImg Method to plot the genes according to their frequency in a microarray like image

---

Description

Generate an image per value of option representing the features (on dot per feature). The color of the dot depends on the frequency of the feature in for the given value of option (number of genes or threshold).

Arguments

`object`	`Object` of `class` `assessment`. Object assessment of interest.
`storagePath`	`character`. URL where the image must be stored.

Methods

object = "assessment" The method is only applicable on objects of class assessment.

Examples

```
## Not run:
data('vV70genesDataset')

expeOfInterest <- new("assessment", dataset=vV70genes,
                                    noFolds1stLayer=3,
```

```
                                    noFolds2ndLayer=2,
                                    classifierName="svm",
                                    typeFoldCreation="original",
                                    svmKernel="linear",
                                    noOfRepeat=10,
                          featureSelectionOptions=new("geneSubsets", optionValues=c(1,2,3,4,5,6)))

expeOfInterest <- runOneLayerExtCV(expeOfInterest)

rankedGenesImg(expeOfInterest, storagePath='myPath')

## End(Not run)
```

`runOneLayerExtCV-methods`

runOneLayerExtCV: Method to run an external one-layer cross-validation

---

Description

This method run an external one-layer cross-validation according to the options stored in an object of class assessment. The concept of external cross-validation has been introduced by G.J. McLachlan and C. Ambroise in 'Selection bias in gene extraction on the basis of microarray gene-expression data' (cf. section References). This technique of cross-validation is used to determine an unbiased estimate of the error rate when feature selection is involved.

Arguments

`object`	`Object of class assessment`. Object assessment of interest

Value

`object of class assessment` in which the one-layer external cross-validation has been computed, therfore, the slot `resultRepeated1LayerCV` is no more NULL. This methods print out the key results of the assessment, to access the full detail of the results, the user must call the method `getResults`.

Methods

object = "assessment" This method is only applicable on objects of class assessment.

References

C. Amboise and G.J. McLachlan 2002. selection bias in gene extraction on the basis of microarray gene-expression data. PNAS, 99(10):6562-6566

See Also

`assessment`, `getResults`, `runTwoLayerExtCV-methods`

Examples

```
data('vV70genesDataset')

# assessment with RFE and SVM
myExpe <- new("assessment", dataset=vV70genes,
                  noFolds1stLayer=9,
                  noFolds2ndLayer=10,
                  classifierName="svm",
                  typeFoldCreation="original",
                  svmKernel="linear",
                  noOfRepeat=2,
               featureSelectionOptions=new("geneSubsets", optionValues=c(1,2,3,4,5,6)))

myExpe <- runOneLayerExtCV(myExpe)
```

`runTwoLayerExtCV-methods`

runTwoLayerExtCV: Method to run an external two-layers cross-validation

---

Description

This method run an external two-layers cross-validation according to the options stored in an object of class assessment. The concept of two-layers cross-validation has been introduced by J.X. Zhu,G.J. McLachlan, L. Ben-Tovim Jonesa, I.A.Wood in 'On selection biases with prediction rules formed from gene expression data' and by I. A. Wood, P. M. Visscher, and K. L. Mengersen in 'Classification based upon gene expression data: bias and precision of error rates' (cf. section References). This technique of cross-validation is used to determine an unbiased estimate of the best error rate (using the best size of subset for RFE-SVM, of the best threshold for NSC) when feature selection is involved.

Arguments

`object`	`Object` of `class` `assessment`. Object assessment of interest

Value

`object` of `class` `assessment` in which the one-layer external cross-validation has been computed, therfore, the slot `resultRepeated2LayerCV` is no more NULL. This methods print out the key results of the assessment, to access the full detail of the results, the user must call the method `getResults`.

Methods

object = "assessment" This method is only applicable on objects of class assessment.

References

J.X. Zhu, G.J. McLachlan, L. Ben-Tovim, I.A. Wood (2008), "On selection biases with prediction rules formed from gene expression data", Journal of Statistical Planning and Inference, 38:374-386.

I.A. Wood, P.M. Visscher, and K.L. Mengersen "Classification based upon gene expression data: bias and precision of error rates" Bioinformatics, June 1, 2007; 23(11): 1363 - 1370.

See Also

assessment, getResults, runOneLayerExtCV-methods

Examples

```
data('vV70genesDataset')

# assessment with RFE and SVM
myExpe <- new("assessment", dataset=vV70genes,
                  noFolds1stLayer=9,
                  noFolds2ndLayer=10,
                  classifierName="svm",
                  typeFoldCreation="original",
                  svmKernel="linear",
                  noOfRepeat=2,
                featureSelectionOptions=new("geneSubsets", optionValues=c(1,2,3,4,5,6)))

myExpe <- runTwoLayerExtCV(myExpe)
```

---

setDataset-methods *getDataset<- Method to modify the attributes of a dataset from an assessment*

Description

This method provides an easy interface to modify the attributes of a dataset directly from an object assessment. The argument topic specifies which part of the dataset should be modified. This method is only available none of the one-layer CV or two-layers CV have been performed and the final classifier has not been determined yet.

Arguments

object	class assessment. Object assessment of interest
topic	character. Optional argument that specifies which attribute of the dataset must be changed, the possible values are dataId (slot dataId of the dataset), dataPath (slot dataPath of the dataset), geneExprFile (slot geneExprFile of the dataset), classesFile (slot classesFile of the dataset), if the topic is missing then the whole dataset object is replaced.
value	The replacement value.

Value

The methods modifies the object of class assessment and returned the slot modified accordingly to the request provided by `topic`.

If 'topic' is missing `object` of `class` `dataset` the dataset corresponding to the assessment is replaced by 'value'.

If 'topic' is "dataId" `object` of `class` `character` the 'dataId' of the dataset is replaced by 'value'

If 'topic' is "dataPath" `object` of `class` `character` the 'dataPath' of the dataset is replaced by 'value'

If 'topic' is "geneExprFile" `object` of `class` `character` the 'geneExprFile' of the dataset is replaced by 'value'

If 'topic' is "classesFile" `object` of `class` `character` the 'classesFile' of the dataset is replaced by 'value'

Methods

object = "assessment" This method is only applicable on objects of class assessment.

Author(s)

Camille Maumet

See Also

`assessment`, `getDataset-methods`

Examples

```
## Not run:
aDataset <- new("dataset", dataId="vantVeer_70", dataPath="pathToFile")
aDataset <- loadData(aDataset)

expeOfInterest <- new("assessment", dataset=aDataset,
                                    noFolds1stLayer=10,
                                    noFolds2ndLayer=9,
                                    classifierName="svm",
                                    typeFoldCreation="original",
                                    svmKernel="linear",
                                    noOfRepeat=2,
                        featureSelectionOptions=new("geneSubsets", optionValues=c(1,2,3,4,5,6)))

# Modify the dataId
getDataset(expeOfInterest, topic='dataId') <- "khan"
getDataset(expeOfInterest, 'dataId')

# Replace the dataset
getDataset(expeOfInterest) <- aDataset
getDataset(expeOfInterest, 'dataId')

## End(Not run)
```

setFeatureSelectionOptions-methods

getFeatureSelectionOptions<- Method to modify the attributes of a featureSelectionOptions from an assessment

Description

This method provides an easy interface to modify the attributes of the object of class featureSelectionOptions related to a particular assessment, directly from this object assessment. The argument `topic` specifies which part of the featureSelectionOptions is of interest. This method is only available none of the one-layer CV or two-layers CV have been performed and the final classifier has not been determined yet.

Arguments

`object`	`Object` of `class` `assessment`. Object assessment of interest
`topic`	`character`. Optional argument that specifies which attribute of the featureSelectionOptions must be replaced, the possible values are: `"optionValues"` (slot `optionValues` of the featureSelectionOptions), `"noOfOptions"` (slot `noOfOptions` of the featureSelectionOptions), if the featureSelectionOptions object is an object of class `geneSubsets`, then the following values are also available for the argument `topic` `"subsetsSizes"` (slot `optionValues` of the geneSubsets), `"noModels"` (slot `noOfOptions` of the geneSubsets), `"maxSubsetSize"` (slot `maxSelectedFeatures` of the geneSubsets), `"speed"` (slot `speed` of the featureSelectionOptions), if the featureSelectionOptions object is an object of class `thresholds`, then the following values are also available for the argument `topic` `"thresholds"` (slot `optionValues` of the object thresholds), `"noThresholds"` (slot `noOfOptions` of the object thresholds) if the `topic` is missing then the whole featureSelectionOptions object is replaced.

Value

The methods modifies the object of class assessment and returned the slot modified accordingly to the request provided by `topic`.

If `topic` is missing `object` of `class` `featureSelectionOptions` featureSelectionOptions corresponding to the assessment is replaced by `value`.

If `topic` is `"optionValues"` `numeric` Slot `optionValues` of the featureSelectionOptions is replaced by `value`.

If `topic` is "noOfOptions" `numeric` Slot `noOfOptions` of the featureSelectionOptions is replaced by `value`.

If `object` is of class `geneSubsets` and `topic` is "maxSubsetSize" `numeric` Slot `maxSubsetSize` of the geneSubsets is replaced by `value`.

If `object` is of class `geneSubsets` and `topic` is "subsetsSizes" `numeric` Slot `optionValues` of the geneSubsets is replaced by `value`.

If `object` is of class `geneSubsets` and `topic` is "noModels" `numericSlot` `noOfOptions` of the geneSubsets is replaced by `value`.

If `object` is of class `geneSubsets` and `topic` is "speed" `numeric` Slot `speed` of the geneSubsets is replaced by `value`.

If `object` is of class `thresholds` and `topic` is "thresholds" `numeric` Slot `optionValues` of the object of class thresholds is replaced by `value`.

If `object` is of class `thresholds` and `topic` is "noThresholds" `numeric` Slot `noOfOptions` of the object of class thresholds is replaced by `value`.

Methods

object = "assessment" The method is only applicable on objects of class assessment.

Author(s)

Camille Maumet

See Also

`featureSelectionOptions`, `assessment`

Examples

```
# With an assessment using RFE
data('vV70genesDataset')

mySubsets <- new("geneSubsets", optionValues=c(1,2,3,4,5,6))
myExpe <- new("assessment", dataset=vV70genes,
                                  noFolds1stLayer=10,
                                  noFolds2ndLayer=9,
                                  classifierName="svm",
                                  typeFoldCreation="original",
                                  svmKernel="linear",
                                  noOfRepeat=2,
                                  featureSelectionOptions=mySubsets)

# Modify the size of the biggest subset
getFeatureSelectionOptions(myExpe, topic='maxSubsetSize') <- 70
getFeatureSelectionOptions(myExpe, topic='maxSubsetSize')
# Modify all the sizes of subsets
getFeatureSelectionOptions(myExpe, topic='subsetsSizes') <- c(1,5,10,25,30)
getFeatureSelectionOptions(myExpe, topic='subsetsSizes')
# Modify the speed
getFeatureSelectionOptions(myExpe, topic='speed') <- 'slow'
getFeatureSelectionOptions(myExpe, topic='speed')
# Modify the entire geneSubsets
getFeatureSelectionOptions(myExpe) <- mySubsets
getFeatureSelectionOptions(myExpe, topic='maxSubsetSize')
getFeatureSelectionOptions(myExpe, topic='subsetsSizes')
getFeatureSelectionOptions(myExpe, topic='speed')
getFeatureSelectionOptions(myExpe, topic='noModels')
```

```
# With an assessment using NSC as a feature selection method
myThresholds <- new("thresholds", optionValues=c(0.1,0.2,0.3))
myExpe2 <- new("assessment", dataset=vV70genes,
                                  noFolds1stLayer=10,
                                  noFolds2ndLayer=9,
                                  classifierName="nsc",
                                  featureSelectionMethod='nsc',
                                  typeFoldCreation="original",
                                  svmKernel="linear",
                                  noOfRepeat=2,
                                  featureSelectionOptions=myThresholds)

otherThresholds <- new("thresholds", optionValues=c(0,0.5,1,1.5,2,2.5,3))
# Modify the whole object 'featureSelectionOptions' (an object of class thresholds)
getFeatureSelectionOptions(myExpe2) <- otherThresholds
getFeatureSelectionOptions(myExpe2, topic='thresholds')
getFeatureSelectionOptions(myExpe2, topic='noThresholds')
```

`show-methods` *show Display the object, by printing, plotting or whatever suits its class*

Description

Implementation of R method show to display object from package `Rmagpie`.

Author(s)

Camille Maumet

`thresholds-class` *thresholds: A class to handle the thresholds to be tested during training of the Nearest Shrunken Centroid*

Description

The Nearest Shrunken Centroid is computed using a threshold. This threshold is usually determined by finding the best threshold value over a set of values by finding the threshold leading to the best error rate assessed by cross-validation. This class stores the values of thresholds to be tried. If the user wants to use default values it's also possible.

Creating objects

`new("thresholds")`

Create an empty thresholds. The default thresholds values will be computed and this object updated as soon as it is linked in an assessment.

`new("thresholds", optionValues)`

Create a thresholds, containing the thresholds values defined by `optionValues`. The slot `noOfOptions` is automatically updated.

Slots

`optionValues`: `numeric` Values of the thresholds, if `optionValues` has length zero then the default thresholds values must be used.

`noOfOptions`: `numeric` Number of thresholds.

Extends

Class "`featureSelectionOptions`", directly.

Methods

`getNoThresholds(thresholds)` Retreive the number of the thresholds (slot noOfOptions)

`getOptionValues(thresholds)`, `getOptionValues(thresholds)<-` Retreive and modify the values of the thresholds (slot optionValues)

Author(s)

Camille Maumet

See Also

`geneSubsets`, `assessment`

Examples

```
# Empty thresholds, the default values will be used when added to an assessment
emptThresholds <- new("thresholds")
getOptionValues(emptThresholds)
getNoThresholds(emptThresholds)

# Another thresholds
thresholds <- new("thresholds", optionValues=c(0,0.1,0.2,1,2))
getOptionValues(thresholds)
getNoThresholds(thresholds)

# Set the thresholds
newThresholds <- c(0.1,0.2,0.5,0.6,1)
getOptionValues(thresholds) <- newThresholds
getOptionValues(thresholds)
getNoThresholds(thresholds)
```

vV70genes *vV70genes: van't Veer et al. 70 best genes in an object of class dataset.*

Description

Gene Expression values and output classes of the 70 best genes selected by van't Veer et al. (cf. references) on 78 patients in 'Gene expression profiling predicts clinical outcome of breast cancer'.

Usage

```
data('vV70genesDataset')
```

Format

lksdjskh

References

L.J. van 't Veer LJ, H. Dai, M.J. van de Vijver, Y.D. He, A.A. Hart, M. Mao, H.L. Peterse,K. van der Kooy, M.J. Marton, A.T. Witteveen, G.J. Schreiber, R.M. Kerkhoven, C. Roberts, P.S. Linsley, R. Bernards, S.H. Friend, 'Gene expression profiling predicts clinical outcome of breast cancer', in Nature. 2002 Jan 31;415(6871):484-5.

Examples

```
data('vV70genesDataset')
vV70genes
```

Index