Package 'cancerclass'

October 31, 2011

Type Package

Version 1.1.0

Date 2011-10-26

Title Development and validation of diagnostic tests from high-dimensional molecular data

Author Jan Budczies, Daniel Kosztyla

Maintainer Daniel Kosztyla <danielkossi@hotmail.com>

Description The classification protocol starts with a feature selection step and continues with nearest-centroid classification. The accurarcy of the predictor can be evaluated using training and test set validation, leave-one-out cross-validation or in a multiple random validation protocol. Methods for calculation and visualization of continuous prediction scores allow to balance sensitivity and specificity and define a cutoff value according to clinical requirements.

Depends R (>= 2.10.1), Biobase, binom, methods, stats

License GPL (>= 2)

biocViews Cancer, Microarray, Classification, Visualization

Collate validation-class.R nvalidation-class.R prediction-class.R predictor-class.R cancerclass-internal.R fit.R loo.R nvalidate.R plot3d_nvalidation.R plot3d_validation.R plot_nvalidation.R plot_prediction.R plot_predictor.R plot_validation.R predict_predictor.R summary_prediction.R validate.R

R topics documented:

`cancerclass-package`

Development and validation of diagnostic tests from high-dimensional molecular data

---

Description

This package implements classification and validation methods for high-dimensional applications, such as gene expression data. The classification protocol starts with a feature selection step and continues with nearest-centroid classification. The accurarcy of the predictor can be evaluated using training and test set validation, leave-one-out cross-validation or in a multiple random validation protocol [1]. Methods for calculation and visualization of continuous prediction scores allow to balance sensitivity and specificity and define a cutoff value according to clinical requirements.

Details

Package:	cancerclass
Type:	Package
Version:	1.1.0
Date:	2010-10-26
License:	GPL (>=2)

Author(s)

Jan Budczies <jan.budczies@charite.de>, Daniel Kosztyla <danielkossi@hotmail.com>

References

[1] Michiels S, Koscielny S, Hill C (2005), *Prediction of cancer outcome with microarrays: a multiple random validation strategy*, Lancet 365:488-492.

See Also

`fit`, `GOLUB`, `GOLUB1`, `loo`, `nvalidate`, `nvalidation-class`, `plot`, `plot,nvalidation-method`, `plot,prediction-method`, `plot,predictor-method`, `plot,validation-method`, `plot3d`, `plot3d,nvalidation-method`, `plot3d,validation-method`, `predict`, `prediction-class`, `predictor-class`, `summary`, `validate`, `validation-class`, `VEER`, `VEER1`, `VIJVER`, `VIJVER1`, `YOUNG`, `YOUNG1`, `cancerclass-internal`, `ilogit`, `calc.roc`, `calc.auc`, `get.d`, `get.d2`, `get.prop`, `get.ntrain`, `prepare`, `filter`

fit *Fitting of a predictor*

Description

Fits a predictor to a training data set.

Usage

```
fit(eset, class="class", method = "welch.test", hparam = 0.75)
```

Arguments

eset	Bioconductor ExpressionSet
class	Character vector specifying classes related to the samples.
method	Character string specifying the feature selection method. Possible values are "cor", "student.test", "welch.test", "wilcoxon.test", "foldchange", "copa", "os", "ort", "shift", "throw".
hparam	Hyperparamter needed for the feature selection methods: Confidential Interval for copa, ort, os (e.g. 0.75, 0.95). Minimum number of samples in each class after applying shift/throw (only necessary for the feature selection methods: throw, shift).

Details

The matrix eset contains the expression signatures of the patients in the columns. The vector class contains the class membership of each sample or patient. Only two-class problems are supported. The colnames of eset are matched to the names of classifier (if both exist).

The hyperparameter hparam describes the minimum number of samples in each class after applying shift/throw. For copa the hyperparameter is quanilte for the definition of outliers. Typical values are 0.75 (default), 0.9, 0.95.

A nearest centroid predictor is constructed by calculating the average level ofeach feature in each of the two classes of the trainig data set.

Value

A predictor object, see predictor.object for details.

See Also

predictor

Examples

```
data(VEER1); data(VIJVER1)
VIJVER2 <- VIJVER1[, setdiff(sampleNames(VIJVER1), sampleNames(VEER1))]
predictor <- fit(VEER1, method="welch.test")
prediction <- predict(predictor, VIJVER2, positive="DM", dist="cor")
plot(prediction, positive="DM", type="histogram", score="zeta")
plot(prediction, positive="DM", type="curves", score="zeta")
plot(prediction, positive="DM", type="roc", score="zeta")
```

GOLUB *GOLUB DATA*

Description

Gene expression data from the leukemia microarray study of Golub et al. [1]. Dataset GOLUB has a dimention of 7129 genes in 72 tumors samples. Dataset GOLUB1 has a dimention of 3571 genes in 72 tumors samples. This dataset is filtered and preprocessed as described in [2].

Usage

```
data(GOLUB)
data(GOLUB1)
```

Value

Data and annotations are organized in a ExtressenSet of the package Biobase.

GOLUB	ExpressionSet (7129 genes in 72 tumors)
GOLUB1	ExpressionSet (3571 genes in 72 tumors)

References

[1] Golub TR et al (1999), *Molecular Classification of cancer: class Discovery and Class Prediction by gene expression monitoring*, Science 286:531-7.
[2] Dudoit S, Fridlyand J (2002), *A prediction-based resampling method for estimating the number of clusters in a dataset*, Genome Biol. 3(7):RESEARCH0036.

Examples

```
###validate
data(GOLUB)
val  <- validate(GOLUB)
plot(val, type="xy")
plot(val, type="genes")
plot(val, type="samples")

###nvalidate
data(GOLUB)
nval  <- nvalidate(GOLUB)
plot(nval, type="xy")
plot(nval, type="genes")
plot(nval, type="samples")

###fit + predict + summary
data(VEER1); data(VIJVER1)
VIJVER2 <- VIJVER1[, setdiff(sampleNames(VIJVER1), sampleNames(VEER1))]
predictor <- fit(VEER1, method="welch.test")
prediction <- predict(predictor, VIJVER2, positive="DM", dist="cor")
plot(prediction, type="histogram", score="zeta")
plot(prediction, type="curves", score="zeta")
plot(prediction, type="roc", score="zeta")
summary(prediction)
```

```
###loo
data(GOLUB)
cv <- loo(GOLUB, positive="ALL", method="welch.test", dist="cor")
plot(cv, type="histogram", score="zeta")
plot(cv, type="curves", score="zeta")
plot(cv, type="roc", score="zeta")
```

`loo` *Leave-one-out cross-validation*

Description

Fitting and validation of a predictor in a leave-one-out protocol.

Usage

```
loo(eset, class="class", method = "welch.test", ngenes=50, dist="cor", hparam
```

Arguments

`eset`	Bioconductor ExpressionSet
`class`	String specifying the column in pData(eset) that contains the class information.
`method`	Specifying the feature selection method. Possible values are "cor", "student.test", "welch.test", "wilcoxon.test", "foldchange", "copa", "os", "ort", "shift", "throw".
`ngenes`	Number of features used for classification.
`dist`	Metric for distance calculation
`hparam`	Hyperparameter needed for some of the feature selection methods. For methods copa, ors and os: Quantile (e.g. 0.75, 0.9, 0.95) used in order to detect outliers. For methods shift and throw: the minimum number of samples in each class after applying shift or throw.
`positive`	One of the two classes. Membership to this class is considered as positive. Needed in order to calculate sensitivity and specificity of the validation.

Details

A leave-one-out cross-validation is performend by calling fit and predict in a loop.

Value

A `pvalidation` object, see `pvalidation.object` for details.

Examples

```
data(GOLUB)
cv <- loo(GOLUB, positive="ALL", ngenes=50, method="welch.test", dist="cor")
plot(cv, type="histogram", score="zeta")
plot(cv, type="samples", score="zeta")
plot(cv, type="curves", score="zeta")
plot(cv, type="roc", score="zeta")
```

`nvalidate` *Classification in a multiple random validation protocol in pependence of the number of features used for predictor construction*

Description

Feature selection and class prediction in a multiple random validation protocol as it was introduced in [1]. Misclassifications rates are calculated for predictors that include different numbers of features.

Usage

```
nvalidate(eset, class="class", ngenes = c(5, 10, 20, 50, 100, 200, 500, 1000), m
```

Arguments

`eset`	Bioconductor ExpressionSet
`class`	Specification of the column in pData(eset) that contains the class information.
`ngenes`	Numerical vector specifying the numbers features that are used for classification.
`method`	Character string specifying the feature selection method. Possible values are "cor", "student.test", "welch.test", "wilcoxon.test", "foldchange", "copa", "os", "ort", "shift", "throw".
`dist`	Character string specifying the method for calculation of the distance between test samples and the centroids. Possible values are "euclidean", "angle", "cor", "center".
`ntrain`	One of the strings "balanced" or "prevalence" or a numeric vector specifying the number of samples of class1 and the number of samples of class2 in the training sets.
`nrep`	The number of repetitions for each training set size.
`hparam`	Hyperparameter needed for some of the feature selection methods: Quantile used for the methods for copa, ort, os (e.g. 0.75, 0.95). Minimum number of samples in each class after applying shift/throw (only necessary for the feature selection methods: throw, shift).

Details

The matrix `exprs(eset)` contains the expression signatures of the patients in the columns. The character vector `pData(eset)[[class]]` contains the class membership of each sample or patient. Only tow-class problems are supported.

The hyperparameter `hparam` describes the minimum number of samples in each class after applying `shift/throw`. For `copa`, `ort` and `os` the hyperparameter specifies the quantile that has to be exceeded in order to consider a sample as an outlier. Typical values are 0.75 (default), 0.9, 0.95.

Validation is implemented in a multiple random validation protocol [1]. For each training set size, `nrep` training sets are randomly drawn from the patients. Features are selected and the centroid is calculated for each of the two classes in feature space. The test samples are classified to the class with the nearest centroid.

Four methods are available for calculation of the distance between test samples and the centroids: euclidean distance, euclidean distance after centering, angle and Pearson correlation. Calculation of distances is executed using the internal function `get.d`.

Feature selection, classification and validation are for predictors that include ngenes features.

Value

A nvalidation object, see nvalidation.object for details. Objects of this class have a method for the function plot.

See Also

nvalidation

Examples

```
data(GOLUB)
nval  <- nvalidate(GOLUB)
plot(nval, type="xy")
plot(nval, type="ngenes")
plot(nval, type="samples")
```

---

nvalidation-class *Class "nvalidation"*

Description

This class of objects is returned by the function nvalidation and represents the validation of a nearest centroid predictor in a random validation protocol. Objects of this class have methods for the function plot.

Objects from the Class

Objects can be created by calls of the form new("nvalidation", ...). describe objects here

Slots

ngenes: Numerical vector containing the numbers features that are used for classification.

method: Character string specifying the feature selection method. Possible values are "cor", "student.test", "welch.test", "wilcoxon.test", "foldchange", "copa", "os", "ort", "shift", "throw".

dist: Character string specifiying the method for calculation of the distance between test samples and the centroids. Possible values are "euclidean", "angle", "cor", "center".

ntrain: The number of samples in the training set.

nrep: The number of repetitions for each training set size.

hparam: Hyperparameter needed for some of the feature selection methods. For methods copa, ors and os: Quantile (e.g. 0.75, 0.9, 0.95) used in order to detect outliers. For methods shift and throw: the minimum number of samples in each class after applying shift or throw.

misclass: A list containing the total, the class1 and the class2 misclassification rates.

nselected: Contains information, how often each of the genes was selected for a predictor.

samples: Numeric matrix containing the classification rates for each of the samples.

classifier: Numerical or character vector containing the class membership of the samples.

fdata: Feature annotations inherited from the ExpressionSet.

Methods

There are three `plot` methods for visualization of the validation results.

See Also

`nvalidate`, `nvalidation`

Examples

```
showClass("nvalidation")
```

`plot` *Plot Method for 'validation, nvalidation, prediction, predictor' Classes*

Description

- class nvalidation:
  Plot of misclassification rates in dependence of the number of features that were used for classification. Total, class1 and class2 misclassification rates including confidence intervals can be plotted separately.
- class prediction:
  Plot methods for continuous predictions scores. Prediction scores are obtained by validation of a predictor in a test set. Four methods for assessment and visualization of predictor performance can be selected by the parameter `type`.
- class predictor:
  Among the three continuous prediction scores (`z`, `zeta` and `ratio`), `zeta` has the special property to be a linear combination of gene expression values. The plot method works only for the prediction score `zeta` and visualizes the contribution of each gene to the score.
- class validation:
  Plots the misclassification rate in dependence of the training set size. Total, class1 and class2 misclassification rates can be plotted including confidence intervals.

Usage

```
plot(x, y, ...)
```

Arguments

`x`	Object of class `nvalidation`.
`y`	missing
`...`	Further arguments directly passed to `plot`.

Methods

x Object of class `validation`

y missing

type Three different kinds of plots can be generated: a xy-plot showing the misclassification rate in dependence of the training set size (type="xy"), a barplot showing the misclassification rates for each of the samples (type="samples"), a barplot showing how often (in %) a gene is included in a predictor (type="genes").

method A character vector specifying the types of misclassification rates to be plotted. Possible types are the names of the classes and all for the total misclassification rate.

anno Only relevant if type="genes": annotation of the features by array probes (anno="probe") or gene symbols (anno="symbol").

sig Vector of numerical values corresponding to the method vector. The numerical values are equal to the confidence level, if equal to NULL, the corresponding confidence interval is not plotted.

xlog A logical value. If TRUE, a logarithmic scale is used for the x-axis.

pos Position of legend specified by a keyword from the list "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center".

ntrain Only relevant if type="samples" or type="genes". Either results for predictors trained in training sets of different sample sizes (ntrain="all"), or results for predictors trained in training sets of the sample size specified by ntrain.

min.percent Only relevant if type="genes". Theshold for selection of the genes that are plotted.

n Only relevant if type="genes": Number of top genes that are plotted.

col Only relevant if type="samples". Color of the boxes for samples of class1 and of class2.

ylim Range of the y-axis.

cex.names Only relevant if type="samples" or type="genes". Scaling factor for the labels of the x-axis.

... Further arguments directly passed to `plot`.

Methods

x Object of class `nvalidation`

y missing

type Three different kinds of plots can be generated: a xy-plot showing the misclassification rate in dependence of the training set size (type="xy"), a barplot showing the misclassification rates for each of the samples (type="samples"), a barplot showing how often (in %) a gene is included in a predictor (type="genes").

method A character vector specifying the types of misclassification rates to be plotted. Possible types are the names of the classes and all for the total misclassification rate.

anno Only relevant if type="genes": annotation of the features by array probes or gene symbols.

sig Vector of numerical values corresponding to the method vector. The numerical values are equal to the confidence level, if equal to NULL, the corresponding confidence interval is not plotted.

xlog A logical value. If TRUE, a logarithmic scale is used for the x-axis.

pos Position of legend specified by a keyword from the list "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center".

ngenes Only relevant if type="samples" or type="genes". Either results for predictors including different number of genes (ngenes="all"), or results for predictors including the number of genes specified by the numeric value ngenes.

min.percent Only relevant if type="genes". Theshold for selection of the genes that are plotted.

n Only relevant if type="genes": Number of top genes that are plotted.

col Only relevant if type="samples". Color of the boxes for samples of class1 and of class2.

ylim Range of the y-axis.

cex.names Only relevant if type="samples" or type="genes". Scaling factor for the labels of the x-axis.

... Further arguments directly passed to `plot`.

Methods

x Object of class `prediction`

y missing

type Four different kinds of plots can be generated: a histogram showing the distribution of the prediction score in class1 and class2 (type="histogram"), a xy-plot showing sensitivity, specificity, positive prediction value (PPV) and negative prediction value (NPV) in dependence on cutoffs for the prediction score (type="curves"), an ROC curve including calculation of the area under the curve (type="roc"), a barplot showing the prediction score for each of the samples (type="samples").

score Specification of the prediction scores used for the plot: score="z", "zeta" or "ratio". If type="roc" a comparative analysis of two or three scores can be done.

breaks.dist Distance of breaks.

ci Only relevant, if type="roc". The method to calculate confidence intervals for sensitivity and specificity. Possibly values are "exact", "ac", "asymptotic", "wilson", "prop.test", "bayes", "logit", "cloglog", "probit", see R package binom for details.

col Only relevant if type="samples". Numerical or character vector of length two specifying the color of symbols for correct and wrong classifications.

curves Only relevant if type="curves". A vector of strings specifying the curves that are included into the plot. Can include "sensitivity", "specificity", "PPV" and "NPV".

col.curves Only relevant if type="curves". A vector of strings corresponding to curves. Specifies the colors of the plot curves.

lty Only relevant if type="roc". Numerical or character vector corresponding to the vector score. Specifies the line types used for the ROC plot.

npoints Only relevant if type="logistic". Number of points to be plotted.

alpha Only relevant if type="logistic". The probability of class membership is estimated by logistic regression. The parameter alpha specifies the confidence level for the confidence interval of this probability.

main Title of the plot.

cex.names Only relevant if type="samples". Scaling factor for the labels of the x-axis.

... Further arguments directly passed to `plot`.

Methods

x Object of class `validation`

y missing

type Currently only type="genes" is supported.

ngenes Number of genes in the predictor.

dist Character string specifying the method for calculation of the distance between test samples and the centroids. Possible values are "euclidean", "angle", "cor", "center".

anno Annotation of the features by array probes (anno="probe") or gene symbols (anno="symbol").

ylab Label of the y-axis.

main Title of the plot.

... Further arguments directly passed to `plot`.

See Also

`validate`, `validation`

Examples

```
###validate
data(GOLUB)
val  <- validate(GOLUB)
plot(val, type="xy")
plot(val, type="genes")
plot(val, type="samples")

###nvalidate
data(GOLUB)
nval  <- nvalidate(GOLUB)
plot(nval, type="xy")
plot(nval, type="genes")
plot(nval, type="samples")

###fit + predict + summary
data(VEER1); data(VIJVER1)
VIJVER2 <- VIJVER1[, setdiff(sampleNames(VIJVER1), sampleNames(VEER1))]
predictor <- fit(VEER1, method="welch.test")
prediction <- predict(predictor, VIJVER2, positive="DM", dist="cor")
plot(prediction, type="histogram", score="zeta")
plot(prediction, type="curves", score="zeta")
plot(prediction, type="roc", score="zeta")
summary(prediction)

###loo
data(GOLUB)
cv <- loo(GOLUB, positive="ALL", method="welch.test", dist="cor")
plot(cv, type="histogram", score="zeta")
plot(cv, type="curves", score="zeta")
plot(cv, type="roc", score="zeta")
```

`plot3d` *Plot3d method for 'validtion and 'nvalidation' classes*

Description

- nvalidation:
  For each gene, the frequency that it is selected is plotted in dependence of the number of genes in the predictor.
- validation:
  For each gene, the frequency that it is selected is plotted in dependence of the training set size.

Usage

```
plot3d(object,...)
```

Arguments

object	Object of class nvalidation or validation
...	Further arguments directly passed to plot.

See Also

plot3d, persp

Examples

```
#nvalidation:
data(GOLUB)
nval  <- nvalidate(GOLUB)
plot3d(nval)

#validation:
data(GOLUB)
val  <- validate(GOLUB)
plot3d(val)
```

---

predict-methods *Predict Method for 'predictor' Class*

Description

Assessment of the performance of a predictor in a test data set.

Usage

```
predict(object="predictor", ...)
```

Arguments

object	Object of class predictor.
...	Further arguments described below.

Value

A prediction object, see prediction.object for details. Objects of this class have a method for the function plot.

Methods

object Object of class `predictor`.

eset Test set, stored in a Bioconductor ExpressionSet object.

positive String refering to one of the two classes. Sensitivity and specificity calculations are carried out with respect to this class.

class Specification of the column in `pData(eset)` that contains the class information. The default value is class="class".

ngenes Number of features used for classification. The default value is ngenes=50.

dist Character string specifiying the method for calculation of the distance between test samples and the centroids. Possible values are "euclidean", "angle", "center" and the default "cor".

Details

The test samples are classified to the class with the nearest centroid. For methods are available for calculation of the distance between test samples and the centroids: Euclidean distance, euclidean distance after centering, angle and Pearson correlation. Calculation of distances is executed using the internal function `get.d`.

See Also

`fit` `prediction` `predictor`

Examples

```
data(VEER1); data(VIJVER1)
VIJVER2 <- VIJVER1[, setdiff(sampleNames(VIJVER1), sampleNames(VEER1))]
predictor <- fit(VEER1, method="welch.test")
prediction <- predict(predictor, VIJVER2, positive="DM", dist="cor")
plot(prediction, type="histogram", score="zeta")
plot(prediction, type="curves", score="zeta")
plot(prediction, type="roc", score="zeta")
summary(prediction)
```

`prediction-class` *Class "prediction"*

Description

An object of this class is returned by the function `fit.predictor` and contains the results of a validation of a predictor on a test set.

Objects from the Class

Objects of this class can be created by `new("prediction", ...)`.

Slots

`type`: String specifying the type of validation: "traintest" for validation in a test set or "loo" for leave-one-out cross-validation.

`predictor`: The predictor that is validated, object of class `predictor`.

`method`: Character string specifying the feature selection method. Possible values are "cor", "student.test", "welch.test", "wilcoxon.test", "foldchange", "copa", "os", "ort", "shift", "throw".

`ngenes`: Numerical vector containing the numbers features that are used for classification.

`dist`: Character string specifiying the method for calculation of the distance between test samples and the centroids. Possible values are "euclidean", "angle", "cor", "center".

`prediction`: Matrix containing the prediction results. Each row represents a sample of the test data set. The first two columns contain the actual and the predicted class membership of the sample. Columns three, four and five contain the scores "z", "zeta" and "ratio". The score "z" is defined as difference of the distance of the sample from the class1 centroid minus the distance of the sample from the class2 centroid divided by the distance of the two centroids. For calculation of the score "zeta", the sample is orthogonally projected to the straight line through the centroids. Then the difference is calculated between the distance from the class1 and the class2 centroid. The score "ratio" is defined as the logarithm of the ratio of the distance of the sample from the class1 centroid divided by the correponding distance from the class2 centroid. For all three prediction scores, the sample is predicted to belong to class1, if the prediction score is negative, while the sample is predicted to belong to class2, if the prediction score is positive.

`cl`: The column in pData(eset) that was used to define the class membership.

`positive`: Character string or number specifying one of the two classes. This information is used for sensitivity and specificity calculations.

See Also

`predict`, `plot`, `fit`

Examples

```
showClass("prediction")
```

---

predictor-class *Class "predictor"*

Description

An object of this class is returned by the function `predict(...)` and represents a nearest centroid predictor learned on a training data set.

Objects from the Class

Objects can be created by the call `new("predictor", ...)`.

Slots

`predictor`: Object of class `"predictor"`

`cl`: Character string or number specifying one of the two classes.

`method`: Character string specifying the feature selection method that was used for predictor construction: Possible values are "cor", "student.test", "welch.test", "wilcoxon.test", "fold-change", "copa", "os", "ort", "shift", "throw".

`hparam`: Hyperparameter needed for some of the feature selection methods. For methods copa, ors and os: Quantile (e.g. 0.75, 0.9, 0.95) used in order to detect outliers. For methods shift and throw: the minimum number of samples in each class after applying shift or throw.

`fdata`: Feature annotations inherited from the training ExpressionSet.

Methods

`plot`A plot method is only avaiable for the score "zeta" that is a linear combination of features.

See Also

See Also as `fit`, or `predictor` for links to other classes

Examples

```
showClass("predictor")
```

---

summary,prediction-methods

Summary Method for 'prediction' Class

Description

Assessment of the performance of a predictor on a test data set.

Usage

```
summary(object="prediction", ...)
```

Arguments

`object`	Object of class `prediction`.
`...`	Further arguments desribed below.

Methods

object Object of class `prediction`.

positive Character string specifying one of the two classes. Sensitivity and specificity calculations are carried out with respect to this class.

See Also

`prediction`

Examples

```
data(VEER1); data(VIJVER1)
VIJVER2 <- VIJVER1[, setdiff(sampleNames(VIJVER1), sampleNames(VEER1))]
predictor <- fit(VEER1, method="welch.test")
prediction <- predict(predictor, VIJVER2, positive="DM", dist="cor")
plot(prediction, type="histogram", score="zeta")
plot(prediction, type="curves", score="zeta")
plot(prediction, type="roc", score="zeta")
summary(prediction)
```

`validate` *Classification in a Multiple Random Validation Protocol in Dependence of the Training Set Size*

Description

Feature selection and class prediction in a multiple random validation protocol. Misclassifications rates are calculated for different sizes of the training set.

Usage

```
validate(eset, class="class", ngenes = 50, method = "welch.test", dist="cor", nt
```

Arguments

`eset`	Bioconductor ExpressionSet
`class`	Specification of the column in `pData(eset)` that contains the class information.
`ngenes`	Numerical vector specifying the numbers features that are used for classification.
`dist`	Character string specifiying the method for calculation of the distance between test samples and the centroids. Possible values are "euclidean", "angle", "cor", "center".
`method`	Character string specifying the feature selection method. Possible values are "cor", "student.test", "welch.test", "wilcoxon.test", "foldchange", "copa", "os", "ort", "shift", "throw".
`ntrain`	One of the strings "balanced" or "prevalence" or a numeric matrix that contains the numbers of training samples of the first class in the in first row and the numbers of training samples of the second class in the second row.
`nrep`	The number of repeated training-test splits for each training set size.
`hparam`	Hyperparameter needed for some of the feature selection methods. For methods copa, ors and os: Quantile (e.g. 0.75, 0.9, 0.95) used in order to detect outliers. For methods shift and throw: the minimum number of samples in each class after applying shift or throw.

Details

The matrix `exprs(eset)` contains the expression signatures of the patients in the columns. The character vector `pData(eset)[[class]]` contains the class membership of each sample or patient. Only two-class problems are supported.

The hyperparameter `hparam` describes the minimum number of samples in each class after applying shift/throw. For `copa`, `ort` and `os` the hyperparameter specifies the quantile that has to be exceeded in order to consider a sample as an outlier. Typical values are 0.75 (default), 0.9, 0.95.

Validation is implemented in a multiple random validation protocol [1]. For each training set size, `nrep` training sets are randomly drawn from the patients. Features are selected and the centroid is calculated for each of the two classes in feature space. The test samples are classified to the class with the nearest centroid.

Four methods are available for calculation of the distance between test samples and the centroids: euclidean distance, centered euclidean distance, angle and Pearson correlation. Calculation of distances is executed using the internal function `get.d`.

The parameter `ntrain` should be equal to one of the strings "balanced" or "prevalence" or a numeric matrix with two rows. For `ntrain = "balanced"`, a balanced layout is used, i.e. half of the training set is chosen from each of the two classes. For `ntrain = "prevalence"` the training sets are balanced according to the prevalence of the two classes in the entire data set. Further, the user can manually specify the sizes of the training sets.

Value

A `validation` object, see `validation.object` for details. Objects of this class have a method for the function `plot`.

References

[1] Michiels S, Koscielny S, Hill C (2005), *Prediction of cancer outcome with microarrays: a multiple random validation strategy*, Lancet 365:488-92.

See Also

`validation`

Examples

```
data(GOLUB1)
val  <- validate(GOLUB1)
plot(val, type="xy")
plot(val, type="genes")
plot(val, type="samples")
```

---

validation-class *Class "validation"*

Description

An object of this class is returned by the function `validate` and represents the validation of a nearest centroid predictor in a random validation protocol.

Objects from the Class

Objects can of this class be created by `new("validation", ...)`.

Slots

`ngenes`: Numerical vector containing the numbers features that are used for classification.

`method`: Character string specifying the feature selection method. Possible values are "cor", "student.test", "welch.test", "wilcoxon.test", "foldchange", "copa", "os", "ort", "shift", "throw".

`dist`: Character string specifiying the method for calculation of the distance between test samples and the centroids. Possible values are "euclidean", "angle", "cor", "center".

`ntrain`: A numeric vector containing the numbers of samples in the training sets.

`nrep`: The number of repetitions for each training set size.

`hparam`: Hyperparameter needed for some of the feature selection methods. For methods copa, ors and os: Quantile (e.g. 0.75, 0.9, 0.95) used in order to detect outliers. For methods shift and throw: the minimum number of samples in each class after applying shift or throw.

`misclass`: A list containing the total, the class1 and the class2 misclassification rates.

`nselected`: Contains information, how often each of the genes was selected for a predictor.

`samples`: Numeric matrix containing the classification rates for each of the samples.

`classifier`: Numerical or character vector containing the class membership of the samples.

`fdata`: Numerical or character vector containing the class membership of the samples.

Methods

There are three `plot` methods for visualization of the validation results.

See Also

`validate`, `validation`

Examples

```
showClass("validation")
```

VEER *Breast cancer gene expression data (van't Veer)*

Description

Gene expression data from the breast cancer microarray study of van't Veer et al. [1]. The data set VEER includes gene expression values of 24481 genes in 78 tumor samples. The data set VEER1 is a filtered version [2] of VEER including gene expression values of 4948 genes in 78 tumor samples).

Usage

```
data(VEER)
data(VEER1)
```

Value

Data and annotations are organized in a ExtressenSet of the package Biobase.

VEER	ExpressionSet
VEER1	ExpressionSet

References

[1] van 't Veer LJ et al. (2002), *Gene expression profiling predicts clinical outcome of breast cancer*, Nature 415:530-536.
[2] Michiels S, Koscielny S, Hill C (2005), *Prediction of cancer outcome with microarrays: a multiple random validation strategy*, Lancet 365:488-492.

Examples

```
###validate
data(GOLUB)
val  <- validate(GOLUB)
plot(val, type="xy")
plot(val, type="genes")
plot(val, type="samples")

###nvalidate
data(GOLUB)
nval  <- nvalidate(GOLUB)
plot(nval, type="xy")
plot(nval, type="genes")
plot(nval, type="samples")

###fit + predict + summary
data(VEER1); data(VIJVER1)
VIJVER2 <- VIJVER1[, setdiff(sampleNames(VIJVER1), sampleNames(VEER1))]
predictor <- fit(VEER1, method="welch.test")
prediction <- predict(predictor, VIJVER2, positive="DM", dist="cor")
plot(prediction, type="histogram", score="zeta")
plot(prediction, type="curves", score="zeta")
plot(prediction, type="roc", score="zeta")
summary(prediction)

###loo
data(GOLUB)
cv <- loo(GOLUB, positive="ALL", method="welch.test", dist="cor")
plot(cv, type="histogram", score="zeta")
plot(cv, type="curves", score="zeta")
plot(cv, type="roc", score="zeta")
```

VIJVER *Breast cancer gene expression data (Vijver)*

Description

Gene expression data from the breast cancer microarray study of Vijver et al. [1]. The data set VIJVER includes expression values of 24481 genes in 295 tumor samples. The data set VIJVER1 is a filtered version of VIJVER [2] including expression values of 4948 genes in 295 tumor samples.

Usage

```
data(VIJVER)
data(VIJVER1)
```

Value

Data and annotations are organized in a ExtressenSet of the package Biobase.

VIJVER	ExpressionSet
VIJVER1	ExpressionSet

References

[1] van de Vijver MJ, He YD, van't Veer LJ, et al. (2002): *A gene-expression signature as a predictor of survival in breast cancer.* N Engl J Med, 347:1999-2009.
[2] Michiels S, Koscielny S, Hill C (2005), *Prediction of cancer outcome with microarrays: a multiple random validation strategy*, Lancet 365:488-493.

Examples

```
###validate
data(GOLUB)
val  <- validate(GOLUB)
plot(val, type="xy")
plot(val, type="genes")
plot(val, type="samples")

###nvalidate
data(GOLUB)
nval  <- nvalidate(GOLUB)
plot(nval, type="xy")
plot(nval, type="genes")
plot(nval, type="samples")

###fit + predict + summary
data(VEER1); data(VIJVER1)
VIJVER2 <- VIJVER1[, setdiff(sampleNames(VIJVER1), sampleNames(VEER1))]
predictor <- fit(VEER1, method="welch.test")
prediction <- predict(predictor, VIJVER2, positive="DM", dist="cor")
plot(prediction, type="histogram", score="zeta")
plot(prediction, type="curves", score="zeta")
plot(prediction, type="roc", score="zeta")
summary(prediction)

###loo
data(GOLUB)
cv <- loo(GOLUB, positive="ALL", method="welch.test", dist="cor")
plot(cv, type="histogram", score="zeta")
plot(cv, type="curves", score="zeta")
plot(cv, type="roc", score="zeta")
```

YOUNG *Breast cancer gene expression data (van't Veer, young patients)*

Description

Gene expression data from the breast cancer microarray study of van't Veer et al. [1]. The data set VEER includes gene expression values of 24481 genes in 19 tumor samples. The data set VEER1 is a filtered version [2] of VEER including gene expression values of 4948 genes in 19 tumor samples).

Usage

```
data(YOUNG)
data(YOUNG1)
```

Value

Data and annotations are organized in a ExtressenSet of the package Biobase.

YOUNG	ExpressionSet
YOUNG1	ExpressionSet

References

[1] van 't Veer LJ et al (2002), *Gene expression profiling predicts clinical outcome of breast cancer*, Nature 415:530-56.
[2] Michiels S, Koscielny S, Hill C (2005), *Prediction of cancer outcome with microarrays: a multiple random validation strategy*, Lancet 365:488-492.

Examples

```
###validate
data(GOLUB)
val  <- validate(GOLUB)
plot(val, type="xy")
plot(val, type="genes")
plot(val, type="samples")

###nvalidate
data(GOLUB)
nval  <- nvalidate(GOLUB)
plot(nval, type="xy")
plot(nval, type="genes")
plot(nval, type="samples")

###fit + predict + summary
data(VEER1); data(VIJVER1)
VIJVER2 <- VIJVER1[, setdiff(sampleNames(VIJVER1), sampleNames(VEER1))]
predictor <- fit(VEER1, method="welch.test")
prediction <- predict(predictor, VIJVER2, positive="DM", dist="cor")
plot(prediction, type="histogram", score="zeta")
plot(prediction, type="curves", score="zeta")
plot(prediction, type="roc", score="zeta")
summary(prediction)
```

```
###loo
data(GOLUB)
cv <- loo(GOLUB, positive="ALL", method="welch.test", dist="cor")
plot(cv, type="histogram", score="zeta")
plot(cv, type="curves", score="zeta")
plot(cv, type="roc", score="zeta")
```

Index