Package 'ClassifyR'

April 9, 2015

Type Package

Title A framework for two-class classification problems, with applications to differential variability and differential distribution testing.

Version 1.0.18

Date 2014-03-13

Author Dario Strbenac, John Ormerod, Graham Mann, Jean Yang

Maintainer Dario Strbenac <dario.strbenac@sydney.edu.au>

VignetteBuilder knitr

biocViews Classification, Survival

Depends R (>= 3.0.3), methods, Biobase, BiocParallel

Imports locfit, ROCR, grid

Suggests limma, edgeR, car, Rmixmod, ggplot2, gridExtra, BiocStyle, pamr, sparsediscrim, PoiClaClu, curatedOvarianData, parathyroidSE, knitr, klaR, gtable, scales

Description The software formalises a framework for classification in R. There are four stages. Data transformation, feature selection, and prediction. The requirements of variable types and names are fixed, but specialised variables for functions can also be provided. The classification framework is wrapped in a driver loop, that reproducibly does a couple of cross-validation schemes. Functions for differential expression, differential variability, and differential distribution are included. Additional functions may be developed by the user, if they have better performing methods.

Collate classes.R utilities.R calcPerformance.R classifyInterface.R DMDselection.R edgeRselection.R errorMap.R fisherDiscriminant.R distribution.R getLocationsAndScales.R KolmogorovSmirnovSelection.R KullbackLeiblerSelection.R leveneSelection.R likelihoodRatioSelection.R limmaSelection.R mixmodels.R naiveBayesKernel.R nearestShrunkenCentroidSelectionInterface.R nearestShrunkenCentroidTrainInterface.R

nearestShrunkenCentroidPredictInterface.R plotFeatureClasses.R
rankPlot.R runTest.R runTests.R selectionPlot.R
subtractFromLocation.R

License GPL-3

R topics documented:

`calcPerformance` *Add Performance Calculations to a ClassifyResult object*

Description

Annotates the results of calling `runTests` with different kinds of performance measures.

Usage

```
  ## S4 method for signature ClassifyResult
calcPerformance(result, performanceType, ...)
```

Arguments

`result`	An object of class `ClassifyResult`.
`performanceType`	Either `"balanced"` or one of the options provided by `performance`.
`...`	Further arguments that may be used by `performance`.

Details

If `runTests` was run in resampling mode, one performance measure is produced for every resampling. If the leave-out mode was used, then the predictions are concatenated, and one performance measure is calcuated for all predictions.

Because ROCR only provides calculations for two-class classification, this function is only suitable for two-class classification performance measures.

Value

An updated `ClassifyResult` object, with new information in the `performance` slot.

Author(s)

Dario Strbenac

Examples

```
predictTable <- data.frame(sample = 1:5,
                           predicted = factor(sample(LETTERS[1:2], 50, replace = TRUE)))
actual <- factor(sample(LETTERS[1:2], 50, replace = TRUE))
result <- ClassifyResult("Example", "Differential Expression",
                         paste("A", 1:10, sep = ), paste("Gene", 1:50, sep = ),
                         list(1:100, 1:100), list(1:5, 6:15),
                         list(predictTable), actual, list("leave", 2))
result <- calcPerformance(result, "balanced")
performance(result)
```

classifyInterface *Interface for PoiClaClu Package's Classify Function*

Description

Passes along all parameters except `verbose`, from the framework to `Classify`.

Usage

```
classifyInterface(..., verbose = 3)
```

Arguments

`...`	All parameters that `Classify` can accept and also `verbose`.
`verbose`	A number between 0 and 3 for the amount of progress messages to give. This function only prints a progress message if the value is 3.

Value

A result list, the same as is returned by `Classify`.

Author(s)

Dario Strbenac

Examples

```
if(require(PoiClaClu))
{
  readCounts <- CountDataSet(n = 100, p = 1000, 2, 5, 1)
 classifyInterface(readCounts[["x"]], readCounts[["y"]], readCounts[["xte"]], verbose = TRUE)
}
```

ClassifyResult *Container for Storing Classification Results*

Description

Contains a table of actual sample classes and predicted classes, the indices of features selected for each fold of each bootstrap resampling or each hold-out classification, and error rates. This class is not intended to be created by the user, but could be used in another package. It is created by `runTests`.

Constructor

```
ClassifyResult(datasetName, classificationName, originalNames, originalFeatures, rankedFeatures,
```

`datasetName` A name associated with the dataset used.

`classificationName` A name associated with the classification.

`originalNames` Sample names.

`originalFeatures` Feature names.

`rankedFeatures` Indices or names of all features, from most to least important.

`chosenFeatures` Indices or names of features selected at each fold.

`predictions` A `list` of `data.frame` containing information about samples, their actual class and predicted class.

`actualClasses` Factor of class of each sample.

`validation` List with first elment being name of the validation scheme, and other elements providing details about scehme.

Summary

A method which summarises the results is available. `result` is a `ClassifyResult` object.

`show(result)`Prints a short summary of what `result` contains.

Accessors

`result` is a `ClassifyResult` object.

`predictions(result)` Returns a `list` of `data.frame`. Each data.frame contains columns `sample`, `predicted`, and `actual`. For hold-out validation, only one data.frame is returned of all of the concatenated predictions.

`actualClasses(result)` Returns a `factor` class labels, one for each sample.

`features(result)` Returns a `list` of `data.frame`. Each data.frame contains columns `sample`, `predicted`, and `actual`.

`performance(result)` Returns a `list` of performance measures. This is empty until `calcPerformance` has been used.

`names(result)` Returns a `character` vector of sample names.

Author(s)

Dario Strbenac

Examples

```
if(require(curatedOvarianData) && require(sparsediscrim))
{
  data(TCGA_eset)
 badOutcome <- which(pData(TCGA_eset)[, "vital_status"] == "deceased" & pData(TCGA_eset)[, "days_to_death"] <= 3
 goodOutcome <- which(pData(TCGA_eset)[, "vital_status"] == "living" & pData(TCGA_eset)[, "days_to_death"] >= 36
  TCGA_eset <- TCGA_eset[, c(badOutcome, goodOutcome)]
 classes <- factor(rep(c("Poor", "Good"), c(length(badOutcome), length(goodOutcome))))
  pData(TCGA_eset)[, "class"] <- classes
 results <- runTests(TCGA_eset, "Ovarian Cancer", "Differential Expression", resamples = 2, folds = 2)
  show(results)
  predictions(results)
  actualClasses(results)
}
```

`distribution` *Get Frequencies of Feature Selection and Sample Errors*

Description

There are two modes. For aggregating feature selection results, the function counts the number of times each feature was selected in all cross validations. For aggregating classification results, the error rate for each sample is calculated. This is useful in identifying outlier samples that are difficult to classify.

Usage

```
  ## S4 method for signature ClassifyResult
distribution(result, type = c("features", "samples"),
                summary = c("density", "frequency"), plot = TRUE, xMax = NULL, ...)
```

Arguments

`result`	An object of class `ClassifyResult`.
`type`	Whether to calculate sample-wise error rate or the number of times a feature was selected.
`summary`	Whether to plot frequencies or densities. If feature distribution is analysed, it will also cause the retured vector to be a decimal representing the percentage.
`plot`	Whether to draw a histogram of the aggregation.
`xMax`	Maximum bin value for histogram to plot.
`...`	Further parameters, such as `colour` and `fill`, passed to `geom_histogram`.

Value

If `type` is "features", a vector as long as the number of features that were chosen at least once containing the number of times the feature was chosen in cross validations. If `type` is "samples", a vector as long as the number of samples, containing the cross validation error rate of the sample.

Author(s)

Dario Strbenac

Examples

```
if(require(curatedOvarianData) && require(sparsediscrim))
{
  data(TCGA_eset)
 badOutcome <- which(pData(TCGA_eset)[, "vital_status"] == "deceased" & pData(TCGA_eset)[, "days_to_death"] <= 3
 goodOutcome <- which(pData(TCGA_eset)[, "vital_status"] == "living" & pData(TCGA_eset)[, "days_to_death"] >= 36
  TCGA_eset <- TCGA_eset[, c(badOutcome, goodOutcome)]
 classes <- factor(rep(c("Poor", "Good"), c(length(badOutcome), length(goodOutcome))))
  pData(TCGA_eset)[, "class"] <- classes
 result <- runTests(TCGA_eset, "Ovarian Cancer", "Differential Expression", resamples = 2, fold = 2)
  sampleDistribution <- distribution(result, "samples", binwidth = 0.1)
  featureDistribution <- distribution(result, "features", binwidth = 1)
  print(head(sampleDistribution))
  print(head(featureDistribution))
}
```

`DMDselection` *Selection of Differential Distributions with Kullback Leibler Distance*

Description

Ranks features by largest Differences in Means/Medians and Deviations and chooses the features which have best resubstitution performance.

Usage

```
  ## S4 method for signature matrix
DMDselection(expression, classes, ...)
  ## S4 method for signature ExpressionSet
DMDselection(expression, trainParams,
                            predictParams, resubstituteParams, ..., verbose = 3)
```

Arguments

`expression`	Either a `matrix` or `ExpressionSet` containing the training data. For a matrix, the rows are features, and the columns are samples.
`classes`	A vector of class labels.
`trainParams`	A container of class `TrainParams` describing the classifier to use for training.
`predictParams`	A container of class `PredictParams` describing how prediction is to be done.
`resubstituteParams`	An object of class `ResubstituteParams` describing the performance measure to consider and the numbers of top features to try for resubstitution classification.
`...`	Variables passed to `getLocationsAndScales`.

verbose — A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

DMD is defined as $|location_1 - location_2| + |scale_1 - scale_2|$.

The subscripts denote the group which the parameter is calculated for.

Value

A list of length 2. The first element has the features ranked from most important to least important. The second element has the features that were selected to be used for classification.

Author(s)

Dario Strbenac

Examples

```
if(require(sparsediscrim))
{
  # First 25 samples are mixtures of two normals. Last 25 samples are one normal.
 genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(50, 5, 1), rnorm(50, 15, 1)))
 genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn) rnorm(100, 9, 3)))
  classes <- factor(rep(c("Poor", "Good"), each = 25))
  DMDselection(genesMatrix, classes,
               trainParams = TrainParams(), predictParams = PredictParams(),
        resubstituteParams = ResubstituteParams(nFeatures = seq(10, 100, 10), performanceType = "balanced", be
}
```

---

edgeRselection *Feature Selection Based on Differential Expression for RNA-seq*

Description

Performs a differential expression analysis between classes and chooses the features which have best resubstitution performance.

Usage

```
  ## S4 method for signature matrix
edgeRselection(expression, classes, ...)
  ## S4 method for signature ExpressionSet
edgeRselection(expression, normFactorsOptions = NULL,
                           dispOptions = NULL, fitOptions = NULL, trainParams,
                           predictParams, resubstituteParams, verbose = 3)
```

Arguments

expression	Either a matrix or ExpressionSet containing the expression values.
classes	A vector of class labels.
normFactorsOptions	A named list of any options to be passed to calcNormFactors.
dispOptions	A named list of any options to be passed to estimateDisp.
fitOptions	A named list of any options to be passed to glmFit.
trainParams	A container of class TrainParams describing the classifier to use for training.
predictParams	A container of class PredictParams describing how prediction is to be done.
resubstituteParams	An object of class ResubstituteParams describing the performance measure to consider and the numbers of top features to try for resubstitution classification.
...	Unused variables from the matrix method passed to the ExpressionSet method.
verbose	A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

The differential expression analysis follows the standard edgeR steps of estimating library size normalisation factors, calculating dispersion, in this case robustly, and then fitting a generalised linear model followed by a likelihood ratio test.

Value

A list of length 2. The first element has the features ranked from most important to least important. The second element has the features that were selected to be used for classification.

Author(s)

Dario Strbenac

References

edgeR: a Bioconductor package for differential expression analysis of digital gene expression data, Mark D. Robinson, Davis McCarthy, and Gordon Smyth, 2010, Bioinformatics, Volume 26 Issue 1, bioinformatics.oxfordjournals.org/content/26/1/139.

Examples

```
if(require(parathyroidSE) && require(sparsediscrim) && require(PoiClaClu))
{
  data(parathyroidGenesSE)
  expression <- assays(parathyroidGenesSE)[[1]]
  DPN <- which(colData(parathyroidGenesSE)[, "treatment"] == "DPN")
  control <- which(colData(parathyroidGenesSE)[, "treatment"] == "Control")
  expression <- expression[, c(control, DPN)]
  classes <- rep(c("Contol", "DPN"), c(length(control), length(DPN)))
```

```
  expression <- expression[rowSums(expression > 10) > 8, ]
  selected <- edgeRselection(expression, classes,
                             trainParams = TrainParams(classifyInterface, TRUE, TRUE),
                             predictParams = PredictParams(function(){}, TRUE, FALSE,
                                      getClasses = function(result) result[["ytehat"]]),
                   resubstituteParams = ResubstituteParams(nFeatures = seq(10, 100, 10), performanceType = "bala
}
```

---

errorMap *Plot a Grid of Sample Error Rates*

Description

A grid of coloured tiles is drawn. There is one column for each sample and one row for each classification result.

Usage

```
  ## S4 method for signature list
errorMap(results,
            errorColours = list(c("#0000FF", "#3F3FFF", "#7F7FFF", "#BFBFFF", "#FFFFFF"),
                                c("#FF0000", "#FF3F3F", "#FF7F7F", "#FFBFBF", "#FFFFFF")),
              classColours = c("blue", "red"), fontSizes = c(24, 16, 12, 12, 12),
            mapHeight = 4, title = "Error Comparison", showLegends = TRUE, xAxisLabel = "Sample Name",
            showXtickLabels = TRUE, showYtickLabels = TRUE, yAxisLabel = "Analysis",
                  legendSize = grid::unit(1, "lines"), plot = TRUE)
```

Arguments

results	A list of ClassifyResult objects.
errorColours	A vector of colours for error levels.
classColours	Either a vector of colours for class levels if both classes should have same colour, or a list of length 2, with each component being a vector of the same length. The vector has the colour gradient for each class.
fontSizes	A vector of length 5. The first number is the size of the title. The second number is the size of the axes titles. The third number is the size of the axes values. The fourth number is the size of the legends' titles. The fifth number is the font size of the legend labels.
mapHeight	Height of the map, relative to the height of the class colour bar.
title	The title to place above the plot.
showLegends	Logical. IF FALSE, the legend is not drawn.
xAxisLabel	The name plotted for the x-axis. NULL suppresses label.
showXtickLabels	Logical. IF FALSE, the x-axis labels are hidden.

showYtickLabels	Logical. IF FALSE, the y-axis labels are hidden.
yAxisLabel	The name plotted for the y-axis. NULL suppresses label.
legendSize	The size of the boxes in the legends.
plot	Logical. IF TRUE, a plot is produced on the current graphics device.

Details

The names of `results` determine the row names that will be in the plot. The length of `errorColours` determines how many bins the error rates will be discretised to.

Value

A plot is produced and a grob is returned that can be saved to a graphics device.

Author(s)

Dario Strbenac

Examples

```
 predicted <- data.frame(sample = sample(10, 100, replace = TRUE),
                         predicted = rep(c("Healthy", "Cancer"), each = 50))
 actual <- factor(rep(c("Healthy", "Cancer"), each = 5))
result1 <- ClassifyResult("Example", "Differential Expression", LETTERS[1:10], LETTERS[10:1], list(1:100), list(
                         list(predicted), actual, list("fold", 100, 5))
 predicted[, "predicted"] <- sample(predicted[, "predicted"])
result2 <- ClassifyResult("Example", "Differential Deviation", LETTERS[1:10], LETTERS[10:1], list(1:100), list(s
                         list(predicted), actual, validation = list("leave", 1))
# wholePlot <- errorMap(list(Gene = result1, Protein = result2)) # Wait for namespace problems to be fixed.
 # ggsave("wholePlot.png", wholePlot)
```

---

fisherDiscriminant *Classification Using Fisher's LDA*

Description

Finds the decision boundary using the training set, and gives predictions for the test set.

Usage

```
  ## S4 method for signature matrix
fisherDiscriminant(expression, classes, test, verbose = 3)
  ## S4 method for signature ExpressionSet
fisherDiscriminant(expression, test, verbose = 3)
```

Arguments

expression	Either a matrix or ExpressionSet containing the training data. For a matrix, the rows are features, and the columns are samples.
classes	A vector of class labels.
test	Either a matrix or ExpressionSet containing the test data.
verbose	A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

Unlike ordinary LDA, Fisher's version does not have assumptions about the normality of the features.

Value

A vector of class predictions, as long as the number of samples in the test data.

Author(s)

Dario Strbenac

Examples

```
trainMatrix <- matrix(rnorm(1000, 8, 2), ncol = 10)
trainMatrix[1:30, 1:5] <- trainMatrix[1:30, 1:5] + 5 # Make first 30 genes D.E.
testMatrix <- matrix(rnorm(1000, 8, 2), ncol = 10)
testMatrix[1:30, 6:10] <- testMatrix[1:30, 6:10] + 5 # Make first 30 genes D.E.
classes <- factor(rep(c("Poor", "Good"), each = 5))
fisherDiscriminant(trainMatrix, classes, testMatrix)
```

---

functionOrList *Union of Functions and List of Functions*

Description

Allows a slot to be either a function or a list of functions.

Author(s)

Dario Strbenac

Examples

```
SelectionParams(limmaSelection)
SelectionParams(list(limmaSelection, leveneSelection))
```

`getLocationsAndScales` *Calculate Location and Scale*

Description

Calculates the location and scale for each feature.

Usage

```
  ## S4 method for signature matrix
getLocationsAndScales(expression, ...)
  ## S4 method for signature ExpressionSet
getLocationsAndScales(expression, location = c("mean", "median"),
                                  scale = c("SD", "MAD", "Qn"))
```

Arguments

`expression`	Either a `matrix` or `ExpressionSet` containing data. For a matrix, the rows are features, and the columns are samples.
`...`	Unused variables from the `matrix` method passed to the `ExpressionSet` method.
`location`	The location to be calculated.
`scale`	The scale to be calculated.

Details

Location can be either "mean" or "median". Scale can be standard deviation, median absolute deviation, or Q_n.

Value

A `list` of length 2. The first element contains the location for every feature. The second element contains the scale for every feature.

Author(s)

Dario Strbenac

References

Qn: http://www.tandfonline.com/doi/pdf/10.1080/01621459.1993.10476408

Examples

```
genesMatrix <- matrix(rnorm(1000, 8, 4), ncol = 10)
getLocationsAndScales(genesMatrix, "median", "MAD")
```

KolmogorovSmirnovSelection *Selection of Differential Distributions with Kolmogorov Smirnov Distance*

Description

Ranks features by largest Kolmogorov Smirnov distance and chooses the features which have best resubstitution performance.

Usage

```
  ## S4 method for signature matrix
KolmogorovSmirnovSelection(expression, classes, ...)
  ## S4 method for signature ExpressionSet
KolmogorovSmirnovSelection(expression, trainParams,
                           predictParams, resubstituteParams, ..., verbose = 3)
```

Arguments

expression	Either a matrix or ExpressionSet containing the training data. For a matrix, the rows are features, and the columns are samples.
classes	A vector of class labels.
trainParams	A container of class TrainParams describing the classifier to use for training.
predictParams	A container of class PredictParams describing how prediction is to be done.
resubstituteParams	An object of class ResubstituteParams describing the performance measure to consider and the numbers of top features to try for resubstitution classification.
...	For the matrix method, variables passed to the ExpressionSet method. For the ExpressionSet method, the options to be passed to function ks.test.
verbose	A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

Features are sorted in order of biggest distance to smallest. The top number of features is used in a classifier, to determine which number of features has the best resubstitution performance.

Value

A list of length 2. The first element has the features ranked from most important to least important. The second element has the features that were selected to be used for classification.

Author(s)

Dario Strbenac

Examples

```
if(require(sparsediscrim))
{
  # First 25 samples are mixtures of two normals. Last 25 samples are one normal.
 genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(50, 5, 1), rnorm(50, 15, 1)))
 genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn) rnorm(100, 9, 3)))
  classes <- factor(rep(c("Poor", "Good"), each = 25))
  KolmogorovSmirnovSelection(genesMatrix, classes,
                             trainParams = TrainParams(), predictParams = PredictParams(),
                    resubstituteParams = ResubstituteParams(nFeatures = seq(10, 100, 10), performanceType = "bala
}
```

KullbackLeiblerSelection

Selection of Differential Distributions with Kullback Leibler Distance

---

Description

Ranks features by largest Kullback Leibler distance and chooses the features which have best resubstitution performance.

Usage

```
  ## S4 method for signature matrix
KullbackLeiblerSelection(expression, classes, ...)
  ## S4 method for signature ExpressionSet
KullbackLeiblerSelection(expression, trainParams,
                           predictParams, resubstituteParams, ..., verbose = 3)
```

Arguments

`expression`	Either a `matrix` or `ExpressionSet` containing the training data. For a matrix, the rows are features, and the columns are samples.
`classes`	A vector of class labels.
`trainParams`	A container of class `TrainParams` describing the classifier to use for training.
`predictParams`	A container of class `PredictParams` describing how prediction is to be done.
`resubstituteParams`	An object of class `ResubstituteParams` describing the performance measure to consider and the numbers of top features to try for resubstitution classification.
`...`	Variables passed to `getLocationsAndScales`.
`verbose`	A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

Symmetric distance is defined as $1/2 * (location_1 - location_2)^2$

Dynamic distance is defined as $|location_1 - location_2| + |scale_1 - scale_2|$

The subscripts denote the group which the parameter is calculated for.

Value

A list of length 2. The first element has the features ranked from most important to least important. The second element has the features that were selected to be used for classification.

Author(s)

Dario Strbenac

Examples

```
if(require(sparsediscrim))
{
  # First 25 samples are mixtures of two normals. Last 25 samples are one normal.
 genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(50, 5, 1), rnorm(50, 15, 1)))
 genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn) rnorm(100, 9, 3)))
  classes <- factor(rep(c("Poor", "Good"), each = 25))
  KullbackLeiblerSelection(genesMatrix, classes,
                           trainParams = TrainParams(), predictParams = PredictParams(),
                  resubstituteParams = ResubstituteParams(nFeatures = seq(10, 100, 10), performanceType = "balan
}
```

---

leveneSelection *Selection of Differential Variability with Levene Statistic*

Description

Ranks features by largest Levene statistic and chooses the features which have best resubstitution performance.

Usage

```
  ## S4 method for signature matrix
leveneSelection(expression, classes, ...)
  ## S4 method for signature ExpressionSet
leveneSelection(expression,
                    trainParams, predictParams, resubstituteParams, verbose = 3)
```

Arguments

expression	Either a matrix or ExpressionSet containing the training data. For a matrix, the rows are features, and the columns are samples.
classes	A vector of class labels.
trainParams	A container of class TrainParams describing the classifier to use for training.
predictParams	A container of class PredictParams describing how prediction is to be done.
resubstituteParams	An object of class ResubstituteParams describing the performance measure to consider and the numbers of top features to try for resubstitution classification.
...	For the matrix method, variables passed to the ExpressionSet method.
verbose	A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

Levene's statistic for unequal variance between groups is a robust version of Bartlett's statistic.

Value

A list of length 2. The first element has the features ranked from most important to least important. The second element has the features that were selected to be used for classification.

Author(s)

Dario Strbenac

Examples

```
if(require(sparsediscrim))
{
  # Samples in one class with differential variability to other class.
  genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(100, 9, 1)))
 genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn) rnorm(100, 9, 4)))
  classes <- factor(rep(c("Poor", "Good"), each = 25))
  leveneSelection(genesMatrix, classes,
                  trainParams = TrainParams(), predictParams = PredictParams(),
                 resubstituteParams = ResubstituteParams(nFeatures = seq(10, 100, 10),
                                     performanceType = "balanced", better = "lower"))
}
```

likelihoodRatioSelection *Selection of Differential Distributions with Likelihood Ratio Statistic*

Description

Ranks features by largest ratio and chooses the features which have the best resubstitution performance.

Usage

```
  ## S4 method for signature matrix
likelihoodRatioSelection(expression, classes, ...)
  ## S4 method for signature ExpressionSet
likelihoodRatioSelection(expression, trainParams, predictParams,
                         resubstituteParams, alternative = c(location = "different", scale = "differer
                                     ..., verbose = 3)
```

Arguments

expression	Either a matrix or ExpressionSet containing the training data. For a matrix, the rows are features, and the columns are samples.
classes	A vector of class labels.
trainParams	A container of class TrainParams describing the classifier to use for training.
predictParams	A container of class PredictParams describing how prediction is to be done.
resubstituteParams	An object of class ResubstituteParams describing the performance measure to consider and the numbers of top features to try for resubstitution classification.
alternative	A vector of length 2. The first element specifies the location of the alternate hypothesis. The second element specifies the scale of the alternate hypothesis. Acceptable values are "same" or "different".
...	Variables passed to getLocationsAndScales.
verbose	A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

Likelihood ratio test of null hypothesis that the location and scale are the same for both groups, and an alternate hypothesis that is specified by parameters. The location and scale of features is calucated by getLocationsAndScales.

Value

A list of length 2. The first element has the features ranked from most important to least important. The second element has the features that were selected to be used for classification.

Author(s)

Dario Strbenac

Examples

```
if(require(sparsediscrim))
{
  # Samples in one class with differential variability to other class.
  genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(100, 9, 1)))
 genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn) rnorm(100, 9, 4)))
  classes <- factor(rep(c("Poor", "Good"), each = 25))
  likelihoodRatioSelection(genesMatrix, classes,
                 trainParams = TrainParams(), predictParams = PredictParams(),
                resubstituteParams = ResubstituteParams(nFeatures = seq(10, 100, 10),
                                    performanceType = "balanced", better = "lower"))
}
```

`limmaSelection` *Selection of Differentially Expressed Features*

Description

Uses a moderated t-test with empirical Bayes shrinkage to select differentially expressed features.

Usage

```
  ## S4 method for signature matrix
limmaSelection(expression, classes, ...)
  ## S4 method for signature ExpressionSet
limmaSelection(expression, trainParams, predictParams,
                                          resubstituteParams, ..., verbose = 3)
```

Arguments

`expression`	Either a `matrix` or `ExpressionSet` containing the training data. For a matrix, the rows are features, and the columns are samples.
`classes`	A vector of class labels.
`trainParams`	A container of class `TrainParams` describing the classifier to use for training.
`predictParams`	A container of class `PredictParams` describing how prediction is to be done.
`resubstituteParams`	An object of class `ResubstituteParams` describing the performance measure to consider and the numbers of top features to try for resubstitution classification.
`...`	For the `matrix` method, variables passed to the `ExpressionSet` method. For the `ExpressionSet` method, extra parameters that are passed to `TrainParams`.
`verbose`	A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

This selection method looks for differential expression. It uses a moderated t-test.

Value

A list of length 2. The first element has the features ranked from most important to least important. The second element has the features that were selected to be used for classification.

Author(s)

Dario Strbenac

References

Limma: linear models for microarray data, Gordon Smyth, 2005, In: Bioinformatics and Computational Biology Solutions using R and Bioconductor, Springer, New York, pages 397-420.

Examples

```
if(require(sparsediscrim))
{
  # Samples in one class with differential expression to other class.
  genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(100, 9, 1)))
  genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn)
                               c(rnorm(75, 9, 1), rnorm(25, 14, 1))))
  classes <- factor(rep(c("Poor", "Good"), each = 25))

  limmaSelection(genesMatrix, classes,
                 trainParams = TrainParams(), predictParams = PredictParams(),
          resubstituteParams = ResubstituteParams(nFeatures = seq(10, 100, 10), performanceType = "balanced",
}
```

---

mixmodels *Selection of Differential Distributions with Mixtures of Normals*

Description

Fits mixtures of normals for every gene, separately for each class.

Usage

```
  ## S4 method for signature matrix
mixModelsTrain(expression, classes, ...)
  ## S4 method for signature ExpressionSet
mixModelsTrain(expression, ..., verbose = 3)
  ## S4 method for signature list,matrix
mixModelsTest(models, test, ...)
  ## S4 method for signature list,ExpressionSet
mixModelsTest(models, test,
        weighted = c("both", "unweighted", "weighted"), minDifference = 0, verbose = 3)
```

Arguments

expression	Either a matrix or ExpressionSet containing the training data. For a matrix, the rows are features, and the columns are samples.
test	Either a matrix or ExpressionSet containing the test data. For a matrix, the rows are features, and the columns are samples.
classes	A vector of class labels.
weighted	In weighted mode, the difference in densities is summed over all features. If unweighted mode, each features's vote is worth the same. To save computational time, both can be calculated simultaneously.
minDifference	The minimum difference in densities for a feature to be allowed to vote. Can be a vector of cutoffs.
...	For the training or testing function with matrix dispatch, arguments passed to the function with ExpressionSet dispatch. For the training function with ExpressionSet dispatch, extra arguments passed to mixmodCluster. The argument nbCluster is mandatory.
models	A list of length 2 of models generated by the training function. The first element has mixture models the same length as the number of features in the expression data for one class. The second element has the same information for the other class.
verbose	A number between 0 and 3 for the amount of progress messages to give. A higher number will produce more messages.

Details

For weighted voting, the difference between probability density, scaled for the number of samples of each class used in training, is summed for all features. For unweighted voting, each feature votes based on the scaled density difference with the same strength.

Value

For mixModelsTrain, a list of trained models of class MixmodCluster. For mixModelsTest, either a factor of predicted classes for the test data, or lists of factors, if both weighted and unweighted voting or a range of minDifference values was provided.

Author(s)

Dario Strbenac

Examples

```
# First 25 samples are mixtures of two normals. Last 25 samples are one normal.
genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(50, 5, 1), rnorm(50, 15, 1)))
genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn) rnorm(100, 9, 3)))
classes <- factor(rep(c("Poor", "Good"), each = 25))
trained <- mixModelsTrain(genesMatrix, classes, nbCluster = 1:3)
mixModelsTest(trained, genesMatrix, minDifference = 1:3)
```

naiveBayesKernel *Classification Using A Bayes Classifier with Kernel Density Estimates*

Description

Kernel density estimates are fitted to the training data and a naive Bayes classifier is used to classify samples in the test data.

Usage

```
  ## S4 method for signature matrix
naiveBayesKernel(expression, classes, ...)
  ## S4 method for signature ExpressionSet
naiveBayesKernel(expression, test,
        weighted = c("both", "unweighted", "weighted"), minDifference = 0, verbose = 3)
```

Arguments

expression	Either a matrix or ExpressionSet containing the training data. For a matrix, the rows are features, and the columns are samples.
classes	A vector of class labels.
...	Unused variables from the matrix method passed to the ExpressionSet method.
test	Either a matrix or ExpressionSet containing the test data.
weighted	In weighted mode, the difference in densities is summed over all features. If unweighted mode, each features's vote is worth the same. To save computational time, both can be calculated simultaneously.
minDifference	The minimum difference in densities for a feature to be allowed to vote. Can be a vector of cutoffs.
verbose	A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

If weighted is TRUE, then a sample's predicted class is the class with the largest sum of density estimates, scaled for the number of samples in the training data of each class. Otherwise, when weighted is FALSE, each feature has an equal vote, and votes for the class with the largest density, scaled for class sizes in the training set.

Value

A vector of class predictions, as long as the number of samples in the test data, or lists of factors, if both weighted and unweighted voting or a range of minDifference values was provided.

Author(s)

Dario Strbenac, John Ormerod

Examples

```
trainMatrix <- matrix(rnorm(1000, 8, 2), ncol = 10)
trainMatrix[1:30, 1:5] <- trainMatrix[1:30, 1:5] + 5 # Make first 30 genes D.E.
testMatrix <- matrix(rnorm(1000, 8, 2), ncol = 10)
testMatrix[1:30, 6:10] <- testMatrix[1:30, 6:10] + 5 # Make first 30 genes D.E.
classes <- factor(rep(c("Poor", "Good"), each = 5))
naiveBayesKernel(trainMatrix, classes, testMatrix)
```

---

nearestShrunkenCentroidPredictInterface

Interface for pamr.predict *Function from* pamr *CRAN Package*

Description

Restructures variables from ClassifyR framework to be compatible with pamr.predict definition.

Usage

```
  ## S4 method for signature pamrtrained,matrix
nearestShrunkenCentroidPredictInterface(trained, test, ...)
  ## S4 method for signature pamrtrained,ExpressionSet
nearestShrunkenCentroidPredictInterface(trained, test, ..., verbose = 3)
```

Arguments

trained	An object of class pamrtrained.
test	Either a matrix or ExpressionSet containing the test data. For a matrix, the rows are features, and the columns are samples.
...	For the function with matrix dispatch, arguments passed to the function with ExpressionSet dispatch. For the function with ExpressionSet dispatch, arguemnts passed to pamr.predict.
verbose	A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

This function is an interface between the ClassifyR framework and pamr.predict.

Value

A factor of predicted classes for the test data.

Author(s)

Dario Strbenac

See Also

pamr.predict for the function that was interfaced to.

Examples

```
if(require(pamr))
{
  # Samples in one class with differential expression to other class.
  genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(100, 9, 1)))
  genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn)
                              c(rnorm(75, 9, 1), rnorm(25, 14, 1))))
  classes <- factor(rep(c("Poor", "Good"), each = 25))

 fit <- nearestShrunkenCentroidTrainInterface(genesMatrix[, c(1:20, 26:45)], classes[c(1:20, 26:45)])
  nearestShrunkenCentroidPredictInterface(fit, genesMatrix[, c(21:25, 46:50)])
}
```

---

nearestShrunkenCentroidSelectionInterface

Interface for pamr.listgenes *Function from* pamr *CRAN Package*

Description

Restructures variables from ClassifyR framework to be compatible with pamr.listgenes definition.

Usage

```
  ## S4 method for signature matrix
nearestShrunkenCentroidSelectionInterface(expression, classes, ...)
  ## S4 method for signature ExpressionSet
nearestShrunkenCentroidSelectionInterface(expression, trained, ..., verbose = 3)
```

Arguments

expression	Either a matrix or ExpressionSet containing the training data. For a matrix, the rows are features, and the columns are samples.
classes	A vector of class labels.
trained	The output of nearestShrunkenCentroidTrainInterface, which is identical to the output of pamr.listgenes.
...	Extra arguments passed to pamr.listgenes
verbose	A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

This function is an interface between the ClassifyR framework and pamr.listgenes.

The set of features chosen is the obtained by considering the range of thresholds provided to nearestShrunkenCentroidTrainInterface and using the threshold that obtains the lowest cross-validation error rate on the training set.

Value

A list of length 2. The first element is empty. The second element has the features that were chosen after applying the threshold.

Author(s)

Dario Strbenac

See Also

pamr.listgenes for the function that was interfaced to.

Examples

```
if(require(pamr))
{
  # Samples in one class with differential expression to other class.
  genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(100, 9, 1)))
  genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn)
                               c(rnorm(75, 9, 1), rnorm(25, 14, 1))))
  classes <- factor(rep(c("Poor", "Good"), each = 25))

  trained <- nearestShrunkenCentroidTrainInterface(genesMatrix, classes)
  nearestShrunkenCentroidSelectionInterface(genesMatrix, classes, trained)[[2]]
}
```

---

nearestShrunkenCentroidTrainInterface

Interface for pamr.train *Function from* pamr *CRAN Package*

Description

Restructures variables from ClassifyR framework to be compatible with pamr.train definition.

Usage

```
  ## S4 method for signature matrix
nearestShrunkenCentroidTrainInterface(expression, classes, ...)
  ## S4 method for signature ExpressionSet
nearestShrunkenCentroidTrainInterface(expression, ..., verbose = 3)
```

Arguments

expression	Either a matrix or ExpressionSet containing the training data. For a matrix, the rows are features, and the columns are samples.
classes	A vector of class labels.
...	Extra arguments passed to pamr.train.
verbose	A number between 0 and 3 for the amount of progress messages to give. This function only prints progress messages if the value is 3.

Details

This function is an interface between the ClassifyR framework and pamr.train.

Value

A list with elements as described in pamr.train.

Author(s)

Dario Strbenac

See Also

pamr.train for the function that was interfaced to.

Examples

```
if(require(pamr))
{
  # Samples in one class with differential expression to other class.
  genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(100, 9, 1)))
  genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn)
                               c(rnorm(75, 9, 1), rnorm(25, 14, 1))))
  classes <- factor(rep(c("Poor", "Good"), each = 25))

  nearestShrunkenCentroidTrainInterface(genesMatrix, classes)
}
```

---

pamrtrained *Trained pamr Object*

Description

Enables dispatching on it.

Summary

A method which summarises the results is available. result is a ClassifyResult object.

show(result)Prints a short summary of what result contains.

Author(s)

Dario Strbenac

Examples

```
genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(100, 9, 1)))
genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn)
                                c(rnorm(75, 9, 1), rnorm(25, 14, 1))))
classes <- factor(rep(c("Poor", "Good"), each = 25))

result <- nearestShrunkenCentroidTrainInterface(genesMatrix, classes)
class(result)
```

---

plotFeatureClasses *Plot Density and Scatterplot for Genes By Class*

Description

Allows the visualisation of genes which were selected by a feature selection method.

Usage

```
  ## S4 method for signature matrix
plotFeatureClasses(expression, classes, ...)
  ## S4 method for signature ExpressionSet
plotFeatureClasses(expression, rows, plot = c("both", "density", "stripchart"),
                   expressionLabel = expression(log[2](expression)), expressionLimits = c(2, 16),
                   fontSizes = c(24, 16, 12, 12, 12), colours = c("blue", "red"))
```

Arguments

expression	Either a matrix or ExpressionSet containing the training data. For a matrix, the rows are features, and the columns are samples.
classes	A vector of class labels.
...	Unused variables from the matrix method passed to the ExpressionSet method.
rows	A vector specifying which rows of the matrix to plot.
plot	Which plots to draw. Can draw either a density plot, stripchart, or both.
expressionLabel	The axis label for the expression axis.
expressionLimits	The minimum and maximum expression values to plot. Set to NULL to use range of data.
fontSizes	A vector of length 5. The first number is the size of the title. The second number is the size of the axes titles. The third number is the size of the axes values. The fourth number is the size of the legends' titles. The fifth number is the font size of the legend labels.
colours	The colours to plot data of each class in.

Value

Plots.

Author(s)

Dario Strbenac

Examples

```
# First 25 samples are mixtures of two normals. Last 25 samples are one normal.
genesMatrix <- sapply(1:25, function(geneColumn) c(rnorm(50, 5, 1), rnorm(50, 15, 1)))
genesMatrix <- cbind(genesMatrix, sapply(1:25, function(geneColumn) rnorm(100, 9, 3)))
classes <- factor(rep(c("Poor", "Good"), each = 25))
chosen <- 1:5 # First five genes in the data were chosen.

# plotFeatureClasses(genesMatrix, classes, chosen, expressionLimits = NULL) Waiting for gridExtra fix.
```

PredictParams *Parameters for Classifier Prediction*

Description

Collects the function to be used for making predictions and any associated parameters.

Constructor

`PredictParams()` Creates a default PredictParams object. This assumes that the object returned by the classifier has a list element named "class".

`PredictParams(predictor, transposeExpression, multipleResults, intermediate = character(0), getClass`
Creates a PredictParams object which stores the function which will do the class prediction and parameters that the function will use.

`predictor` A `function` to make predictions with. The first argument must accept the classifier made in the training step. The second argument must accept a `matrix` of new data.

`transposeExpression` Set to `TRUE` if `classifier` expects features as columns.

`multipleResults` If `TRUE`, `predictor` will return a `list` of results, perhaps for different values of some parameter. If FALSE, a single result is returned.

`intermediate` Character vector. Names of any variables created in prior stages by `runTest` that need to be passed to the prediction function.

`getClasses` A `function` to extract the vector of class predictions from the result object created by `predictor`.

`...` Other arguments that `predictor` may use.

Author(s)

Dario Strbenac

Examples

```
predictParams <- PredictParams(predictor = predict, TRUE, FALSE, getClasses = function(result) result)
# For prediction by trained object created by dlda function.
PredictParams(predictor = function(){}, TRUE, FALSE, getClasses = function(result) result)
# For when the training function also does prediction and directly returns vector of predictions.
```

---

rankPlot *Plot Pair-wise Overlap of Ranked Features*

Description

The average pair-wise overlap is computed for every pair of cross-validations. The overlap is converted to a percentage and plotted as lineplots.

Usage

```
  ## S4 method for signature list
rankPlot(results, topRanked = seq(10, 100, 10),
                  comparison = c("within", "classificationName", "validation", "datasetName"),
                  lineColourVariable = c("validation", "datasetName", "classificationName", "None"),
                  pointTypeVariable = c("datasetName", "classificationName", "validation", "None"),
                  rowVariable = c("None", "datasetName", "classificationName", "validation"),
                  columnVariable = c("classificationName", "datasetName", "validation", "None"),
                  yMax = 100, fontSizes = c(24, 16, 12, 12, 12), title = "Feature Ranking Stability",
                  xLabelPositions = seq(10, 100, 10), yLabel = "Average Pairwise Common Features (%)",
                           plot = TRUE, parallelParams = bpparam())
```

Arguments

results	A list of ClassifyResult objects.
topRanked	A sequence of thresholds of number of the best features to use for overlapping.
comparison	The aspect of the experimental design to compare. See Details section for a detailed description.
lineColourVariable	The slot name that different levels of are plotted as different line colours.
pointTypeVariable	The slot name that different levels of are plotted as different point shapes on the lines.
rowVariable	The slot name that different levels of are plotted as separate rows of lineplots.
columnVariable	The slot name that different levels of are plotted as separate columns of lineplots.
yMax	The maximum value of the percentage to plot.
fontSizes	A vector of length 5. The first number is the size of the title. The second number is the size of the axes titles. The third number is the size of the axes values. The fourth number is the size of the legends' titles. The fifth number is the font size of the legend labels.

title	An overall title for the plot.
xLabelPositions	Locations where to put labels on the x-axis.
yLabel	Label to be used for the y-axis of overlap percentages.
plot	Logical. IF TRUE, a plot is produced on the current graphics device.
parallelParams	An object of class MulticoreParam or SnowParam.

Details

Possible values for slot names are "datasetName", "classificationName", and "validation". If "None", then that graphic element is not used.

If comparison is "within", then the feature rankings are compared within a particular analysis. The result will inform how stable the feature rankings are between different iterations of a particular analysis. If comparison is "classificationName", then the feature rankings are compared across different classification algorithm types, for each level of "datasetName" and "validation". The result will inform how stable the feature rankings are between different classification algorithms, for every cross-validation scheme and every dataset. If comparison is "validation", then the feature rankings are compared across different cross-validation schemes, for each level of "classificationName" and "datasetName". The result will inform how stable the feature rankings are between different cross-validation schemes, for every classification algorithm and every dataset. If comparison is "datasetName", then the feature rankings are compared across different datasets, for each level of "classificationName" and "validation". The result will inform how stable the feature rankings are between different datasets, for every classification algorithm and every dataset. This could be used to consider if different studies have a highly overlapping feature ranking pattern.

Calculating all pair-wise set overlaps can be time-consuming. This stage can be done on multiple CPUs by providing the relevant options to parallelParams.

Value

An object of class ggplot and a plot on the current graphics device, if plot is TRUE.

Author(s)

Dario Strbenac

Examples

```
 predicted <- data.frame(sample = sample(10, 100, replace = TRUE),
                         predicted = rep(c("Healthy", "Cancer"), each = 50))
 actual <- factor(rep(c("Healthy", "Cancer"), each = 5))
result1 <- ClassifyResult("Example", "Differential Expression", LETTERS[1:10], LETTERS[10:1], list(1:100, c(1:9,
                         list(predicted), actual, list("fold", 100, 5))
 predicted[, "predicted"] <- sample(predicted[, "predicted"])
result2 <- ClassifyResult("Example", "Differential Variability", LETTERS[1:10], LETTERS[10:1], list(1:100, c(1:5
                         list(predicted), actual, validation = list("leave", 1))
# rankPlot(list(result1, result2), pointTypeVariable = "classificationName") # Wait for namespace problems to be
```

ResubstituteParams *Parameters for Resubstitution Error Calculation*

Description

Some feature selection functions provided in the framework use resubstitution error rate to choose the best number of features for classification. This class stores parameters related to that process

Constructor

`ResubstituteParams()` Creates a default ResubstituteParams object. The number of features tried is 100, 200, 300, 400, 500. The performance measure used is the balanced error rate.

`ResubstituteParams(nFeatures, preformanceType, better = c("lower", "higher"))` Creates a ResubstituteParams object, storing information about the number of top features to calculate the performance measure for, the performance measure to use, and if higher or lower values of the measure are better.

`nFeatures` A vector for the top number of features to test the resubstitution error for.

`performanceType` Either `"balanced"` or one of the options provided by `performance`.

`better` Either `"lower"` or `"higher"`. Determines whether higher or lower values of the performance measure are desirable.

`intermediate` Character vector. Names of any variables created in prior stages by `runTest` that need to be passed to `classifier`.

`...` Other named parameters which will be used by the classifier.

Author(s)

Dario Strbenac

Examples

```
ResubstituteParams(nFeatures = seq(25, 1000, 25), performanceType = "err", better = "lower")
```

runTest *Perform a Single Classification*

Description

For a dataset of features and samples, the classification process is run. It consists of data transformation, feature selection, training and testing.

Usage

```
  ## S4 method for signature matrix
runTest(expression, classes, ...)
  ## S4 method for signature ExpressionSet
runTest(expression,
              training, testing, params = list(SelectionParams(), TrainParams(), PredictParams()),
                     verbose = 1)
```

Arguments

expression	Either a matrix or ExpressionSet containing the training data. For a matrix, the rows are features, and the columns are samples.
classes	A vector of class labels.
training	A vector which specifies the training samples.
testing	A vector which specifies the test samples.
params	A list of objects of class of TransformParams, SelectionParams, TrainParams, or PredictParams. The order they are in the list determines the order in which the stages of classification are done in.
...	Unused variables from the matrix method passed to the ExpressionSet method.
verbose	A number between 0 and 3 for the amount of progress messages to give. A higher number will produce more messages.

Details

This function only performs one classification and prediction. See runTests for a driver function that does cross validation and uses this function.

Value

A list with three elements. The first element contains the indices of genes that were selected by the feature selection step. The second element contains the indices of the samples that were in the test set. The third element contains a vector of the classes predicted by the classifer.

Author(s)

Dario Strbenac

Examples

```
if(require(curatedOvarianData) && require(sparsediscrim))
{
  data(TCGA_eset)
 badOutcome <- which(pData(TCGA_eset)[, "vital_status"] == "deceased" & pData(TCGA_eset)[, "days_to_death"] <= 3
 goodOutcome <- which(pData(TCGA_eset)[, "vital_status"] == "living" & pData(TCGA_eset)[, "days_to_death"] >= 36
  TCGA_eset <- TCGA_eset[, c(badOutcome, goodOutcome)]
 classes <- factor(rep(c("Poor", "Good"), c(length(badOutcome), length(goodOutcome))))
  pData(TCGA_eset)[, "class"] <- classes
  runTest(TCGA_eset, training = (1:ncol(TCGA_eset)) %% 2 == 0,
```

```
            testing = (1:ncol(TCGA_eset)) %% 2 != 0)
}
```

---

runTests *Reproducibly Do Resampling or Leave Out and Cross Validation*

Description

Enables doing classification schemes such as 100 resamples 5-fold cross validation or leave one out cross validaion. Processing in parallel is possible by leveraging the package BiocParallel.

Usage

```
  ## S4 method for signature matrix
runTests(expression, classes, ...)
  ## S4 method for signature ExpressionSet
runTests(expression, datasetName, classificationName,
              validation = c("bootstrap", "leaveOut"), bootMode = c("fold", "split"),
              resamples = 100, percent = 25, folds = 5, leave = 2, seed, parallelParams = bpparam(),
               params = list(SelectionParams(), TrainParams(), PredictParams()),
                     verbose = 1)
```

Arguments

expression	Either a matrix or ExpressionSet containing the training data. For a matrix, the rows are features, and the columns are samples.
classes	A vector the same length as the number of columns of expression data specifying the class that the samples belong to.
datasetName	A name associated with the dataset used.
classificationName	A name associated with the classification.
validation	"bootstrap" for repeated resampling or "leaveOut" for leaving all combinations of k samples as test samples.
bootMode	Character. Either "fold" or "split". If "fold", then the samples are split into folds and in each iteration one is used as the test set. If "split", the samples are split into two groups. One is used as the training set, the other is the test set.
resamples	Relevant when repeated resampling is used. The number of times to do sampling with replacement.
percent	Used when bootstrap resampling with split method is chosen. The percentage of samples to be in the test set.
folds	Relevant when repeated resampling is used with fold mode. The number of folds to break each resampling into. Each fold is used once as the test set.
leave	Relevant when leave k out validation is used. The number of samples to leave for testing.

seed	The random number generator used for repeated resampling will use this seed, if it is provided. Allows reproducibility of repeated usage on the same input data.
parallelParams	An object of class MulticoreParam or SnowParam.
params	A list of objects of class of TransformParams, SelectionParams, TrainParams, or PredictParams. The order they are in the list determines the order in which the stages of classification are done in.
...	Unused variables from the matrix method passed to the ExpressionSet method.
verbose	A number between 0 and 3 for the amount of progress messages to give. A higher number will produce more messages.

Value

If the predictor function made a single prediction, then an object of class ClassifyResult. If the predictor function made a set of predictions, then a list of such objects.

Author(s)

Dario Strbenac

Examples

```
if(require(curatedOvarianData) && require(sparsediscrim))
{
  data(TCGA_eset)
 badOutcome <- which(pData(TCGA_eset)[, "vital_status"] == "deceased" & pData(TCGA_eset)[, "days_to_death"] <= 3
 goodOutcome <- which(pData(TCGA_eset)[, "vital_status"] == "living" & pData(TCGA_eset)[, "days_to_death"] >= 36
  TCGA_eset <- TCGA_eset[, c(badOutcome, goodOutcome)]
 classes <- factor(rep(c("Poor", "Good"), c(length(badOutcome), length(goodOutcome))))
  pData(TCGA_eset)[, "class"] <- classes
 runTests(TCGA_eset, "Ovarian Cancer", "Differential Expression", resamples = 2, fold = 2)
}
```

SelectionParams *Parameters for Feature Selection*

Description

Collects and checks necessary parameters required for feature selection. The empty constructor is provided for convenience.

Constructor

`SelectionParams()` Creates a default SelectionParams object. This uses a limma t-test and tries 100, 200, 300, 400, 500 features, and picks the number of features with the best resubstitution error rate. Users should create an appropriate `SelectionParams` object for the characteristics of their data, once they are familiar with this software.

`SelectionParams(featureSelection, minPresence = 1, intermediate = character(0),` `subsetExp`
Creates a SelectionParams object which stores the function which will do the selection and parameters that the function will use.

`featureSelection` Either a function which will do the selection or a list of such functions. For a particular function, the first argument must be an `ExpressionSet` object. The function's return value must be a vector of row indices of genes that were selected.

`minPresence` If a list of functions was provided, how many of those must a feature have been selected by to be used in classification. 1 is equivalent to a set union and a number the same length as `featureSelection` is equivalent to set intersection.

`intermediate` Character vector. Names of any variables created in prior stages by `runTest` that need to be passed to a feature selection function.

`subsetExpressionData` Whether to subset the expression data, after selection has been done.

`...` Other named parameters which will be used by the selection function. If `featureSelection` was a list of functions, this must be a list of lists, as long as `featureSelection`.

Author(s)

Dario Strbenac

Examples

```
SelectionParams(limmaSelection, nFeatures = c(25, 50, 75, seq(100, 1000, 100)))

# For pamr shrinkage selection.
SelectionParams(nearestShrunkenCentroidSelectionInterface, intermediate = "trained",
                subsetExpressionData = FALSE)
```

`selectionPlot` *Plot Pair-wise Overlap of Selected Features*

Description

The average pair-wise overlap is computed for every pair of cross-validations. The overlap is converted to a percentage and plotted as a set of boxplots.

Usage

```
  ## S4 method for signature list
selectionPlot(results,
              comparison = c("within", "classificationName", "validation", "datasetName"),
                xVariable = c("classificationName", "datasetName", "validation"),
              boxFillColouring = c("classificationName", "datasetName", "validation", "None"),
                    boxFillColours = NULL,
              boxLineColouring = c("validation", "classificationName", "validation", "None"),
                    boxLineColours = NULL,
              rowVariable = c("None", "validation", "datasetName", "classificationName"),
              columnVariable = c("datasetName", "classificationName", "validation", "None"),
              yMax = 100, fontSizes = c(24, 16, 12), title = "Feature Selection Stability",
              xLabel = "Analysis", yLabel = "Average Pairwise Common Features (%)", margin = grid::unit(c
```

Arguments

results	A list of ClassifyResult objects.
comparison	The aspect of the experimental design to compare. See Details section for a detailed description.
xVariable	The factor to make separate boxes in the boxplot for.
boxFillColouring	A factor to colour the boxes by.
boxFillColours	A vector of colours, one for each level of boxFillColouring.
boxLineColouring	A factor to colour the box lines by.
boxLineColours	A vector of colours, one for each level of boxLineColouring.
rowVariable	The slot name that different levels of are plotted as separate rows of boxplots.
columnVariable	The slot name that different levels of are plotted as separate columns of boxplots.
yMax	The maximum value of the percentage to plot.
fontSizes	A vector of length 5. The first number is the size of the title. The second number is the size of the axes titles. The third number is the size of the axes values. The fourth number is the size of the legends' titles. The fifth number is the font size of the legend labels.
title	An overall title for the plot.
xLabel	Label to be used for the x-axis.
yLabel	Label to be used for the y-axis of overlap percentages.
margin	The margin to have around the plot.
rotate90	Logical. IF TRUE, the boxplot is horizontal.
plot	Logical. IF TRUE, a plot is produced on the current graphics device.
parallelParams	An object of class MulticoreParam or SnowParam.

Details

Possible values for slot names are "datasetName", "classificationName", and "validation". If "None", then that graphic element is not used.

Calculating all pair-wise set overlaps can be time-consuming. This stage can be done on multiple CPUs by providing the relevant options to parallelParams.

Value

An object of class ggplot and a plot on the current graphics device, if plot is TRUE.

Author(s)

Dario Strbenac

Examples

```
predicted <- data.frame(sample = sample(10, 100, replace = TRUE),
                        predicted = rep(c("Healthy", "Cancer"), each = 50))
actual <- factor(rep(c("Healthy", "Cancer"), each = 5))
result1 <- ClassifyResult("Example", "Differential Expression", LETTERS[1:10], LETTERS[10:1], list(1:100, c(1:9,
                        list(predicted), actual, list("fold", 100, 5))
predicted[, "predicted"] <- sample(predicted[, "predicted"])
result2 <- ClassifyResult("Example", "Differential Variability", LETTERS[1:10], LETTERS[10:1], list(1:100, c(1:5
                        list(predicted), actual, validation = list("leave", 1))
# selectionPlot(list(result1, result2), xVariable = "classificationName", xLabel = "Analysis", columnVariable = '
```

---

subtractFromLocation *Subtract All Feature Measurements from Location*

Description

For each feature, calculates the location, and subtracts all measurements from that location.

Usage

```
  ## S4 method for signature matrix
subtractFromLocation(expression, ...)
  ## S4 method for signature ExpressionSet
subtractFromLocation(expression, training, location = c("mean", "median"),
                                 verbose = 3)
```

Arguments

expression	Either a matrix or ExpressionSet containing the data. For a matrix, the rows are features, and the columns are samples.
...	Unused variables from the matrix method passed to the ExpressionSet method.
training	A vector specifying which samples are in the training set.
location	Character. Either "mean" or "median".
verbose	A number between 0 and 3 for the amount of progress messages to give. A higher number will produce more messages.

Details

Only the samples specified by training are used in the calculation of the location. To use all samples for calculation of the location, simply provide indices of all the samples.

Value

An ExpressionSet of the same dimension that was input, with values subtracted from the location specified.

Author(s)

Dario Strbenac

Examples

```
subtractFromLocation(matrix(1:100, ncol = 10), training = 1:5, "median")
```

`TrainParams` *Parameters for Classifier Training*

Description

Collects and checks necessary parameters required for classifier training. The empty constructor is provided for convenience.

Constructor

`TrainParams()` Creates a default TrainParams object. The classifier function is DLDA. Users should create an appropriate `TrainParams` object for the characteristics of their data, once they are familiar with this software.

`TrainParams(classifier, transposeExpression, doesTests, ...)` Creates a TrainParams object which stores the function which will do the classifier building and parameters that the function will use.

`classifier` A function which will construct a classifier, and also possibly make the predictions. The first argument must be a `matrix` object. The second argument must be a vector of classes. The third argument must be `verbose`. If doesTests is `TRUE`, the third argument must be a `matrix` of test data and the fourth argument is `verbose`. The function's return value can be either a trained classifier when `doesTests` is `FALSE` or a vector of class predictions if `doesTests` is `TRUE`.

`transposeExpression` Set to `TRUE` if `classifier` expects features as columns.

`doesTests` Set to `TRUE` if `classifier` also performs and returns predictions.

`intermediate` Character vector. Names of any variables created in prior stages by `runTest` that need to be passed to `classifier`.

`...` Other named parameters which will be used by the classifier.

Author(s)

Dario Strbenac

Examples

```
if(require(sparsediscrim))
  trainParams <- TrainParams(dlda, transposeExpression = TRUE, doesTests = FALSE)
# sparsediscrim has a separate predict method for trained DLDA objects.
# dlda expects features in columns, and samples in rows.
```

TransformParams *Parameters for Data Transformation*

Description

Collects and checks necessary parameters required for transformation. The empty constructor is for when no data transformation is desired. One data transformation function is distributed. See subtractFromLocation.

Constructor

`TransformParams(transform, intermediate = character(0), ...)` Creates a TransformParams object which stores the function which will do the transformation and parameters that the function will use.

`transform` A function which will do the transformation. The first argument must be an ExpressionSet object.

`intermediate` Character vector. Names of any variables created in prior stages by runTest that need to be passed to a feature selection function.

`...` Other named parameters which will be used by the transformation function.

Author(s)

Dario Strbenac

Examples

```
transforParams <- TransformParams(subtractFromLocation, location = "median")
# Subtract all values from training set median, to obtain absolute deviations.
```

Index