# MCRestimate

April 19, 2010

---

`class.factor.format`
*A function for creating a factor from the phenoData slot of an exprSet*

---

## Description

This function creates a factor whose levels represent the different classes for a classification problem. It is derived from the column specified in the argument `class.column`. If the factor has more than two levels, the argument `reference.class` can be used to transform this into a two-class problem: reference class versus the rest.

## Usage

```
class.factor.format(x, class.column, reference.class=NULL)
```

## Arguments

| | |
|---|---|
| `x` | an `exprSet` |
| `class.column` | either a number or a character string specifying the relevant column of the phenoData slot in `x` |
| `reference.class` | character vector. If specified the result will be a factor with only 2 levels: the reference class(es) versus all other |

## Value

A factor of length nrow(pData(x))

## Author(s)

Markus Ruschhaupt `mailto:m.ruschhaupt@dkfz.de`

## Examples

```
library(MCRestimate)
 library(golubEsets)
data(Golub_Train)
class.factor.format(Golub_Train[,28:35], "FAB",reference="M1")
```

`ClassifierBuild` *Building a classifier as a combination of preprocessing and classification method*

## Description

builds a classifier as a combination of preprocessing and classification methods

## Usage

```
ClassifierBuild(eset,
                class.column,
                reference.class=NULL,
                classification.fun,
                variableSel.fun ="identity",
                cluster.fun ="identity",
                poss.parameters=list(),
                cross.inner=10,
                rand=123,
                information=TRUE,
                thePreprocessingMethods=c(variableSel.fun,cluster.fun))
```

## Arguments

| | |
|---|---|
| `eset` | an object of class `exprSet` or `exprSetRG` |
| `class.column` | a number or a character string which indicated the column of the expression set's phenodata containing the class label |
| `reference.class` | a character string with the name of one class - if specified the class will form the first class and all the other classes will form the second class |
| `classification.fun` | a character string which names the function that should be used for the classification |
| `variableSel.fun` | character string which names the function that should be used for variable selection |
| `cluster.fun` | character string which names the function that should be used for clustering the variables |
| `thePreprocessingMethods` | vector of character with the names of all preprocessing functions- can be used instead of 'variableSel.fun' and 'cluster.fun' - see details |
| `poss.parameters` | a list of possible values for the parameter of the classification method |
| `cross.inner` | integer - the number of nearly equal sized parts the train set should be divided into |
| `rand` | integer - the random number generator will be put in a reproducible state |
| `information` | information - should classificator specific data be given(depends on the wrapper for the classification method) |

## Value

a `list` with the following arguments:

`classifier.for.matrix`

`classifier.for.exprSet`

| | |
|---|---|
| `parameter` | a list consisting of the estimated 'best' parameter for each cross-validation part |
| `class.method` | string which names the function used for the classification |
| `thePreprocessingMethods` | character string - name of the preprocessing functions that have been used |
| `cross.inner` | number of blocks for a the inner cross-validation |
| `information` | classificator specific data |

## Author(s)

Markus Ruschhaupt `mailto:m.ruschhaupt@dkfz.de`

## Examples

```
library(MCRestimate)
library(golubEsets)
data(Golub_Train)

class.column <- "ALL.AML"
Preprocessingfunctions <- c("varSel.highest.var")
list.of.poss.parameter <- list(var.numbers = c(250,1000))
classification.funct <- "RF.wrap"
cross.inner <- 5

RF.classifier <- ClassifierBuild(Golub_Train,
     class.column,
classification.fun = classification.funct,
     thePreprocessingMethods = Preprocessingfunctions,
     poss.parameters = list.of.poss.parameter,
     cross.inner = cross.inner)
```

---

# `varSel.highest.var` *Variable selection and cluster functions*

---

## Description

Different functions for a variable selection and clustering methods. These functions are mainly used for the function `MCRestimate`

## Usage

```
identity(sample.gene.matrix,classfactor,...)
       varSel.highest.t.stat(sample.gene.matrix,classfactor,theParameter=NULL,va

       varSel.highest.var(sample.gene.matrix,classfactor,theParameter=NULL,var.n

       varSel.AUC(sample.gene.matrix, classfactor, theParameter=NULL,var.numbers
       cluster.kmeans.mean(sample.gene.matrix,classfactor,theParameter=NULL,numb

       varSel.removeManyNA(sample.gene.matrix,classfactor, theParameter=NULL, NA
       varSel.impute.NA(sample.gene.matrix ,classfactor,theParameter=NULL,...)
```

## Arguments

| | |
|---|---|
| sample.gene.matrix | a matrix in which the rows corresponds to genes and the colums corresponds to samples |
| classfactor | a factor containing the values that should be predicted |
| theParameter | Parameter that depends on the function. For 'cluster.kmeans.mean' eighter NULL or an output of the function kmeans. If it is NULL then kmeans will be used to form clusters of the genes. Otherwise the already existing clusters will be used. In both ways there will be a calculation of the metagene intensities afterwards. For the other functions eighter NULL or a logical vector which indicates for every gene if it sould be left out from further analysis or not |
| number.clusters | parameter which specifies the number of clusters |
| var.numbers | some methods needs an argument which specifies how many variables should be taken |
| NAthreshold | integer- if the percentage of the NA is higher than this threshold the variable will be deleted |
| ... | Further parameters |

## Details

metagene.kmeans.mean performes a kmeans clustering with a number of clusters specified by 'number clusters' and takes the mean of each cluster. varSel.highest.var selects a number (specified by 'var.numbers') of variables with the highest variance. varSel.AUC chooses the most discriminating variables due to the AUC criterium (the library ROC is required).

## Value

Every function returns a list consisting of two arguments:

| | |
|---|---|
| matrix | the result matrix of the variable redution or the clustering |
| parameter | The parameter which are used to reproduce the algorithm, i.e. a vector which indicates for every gene if it will be left out from further analysis or not if a gene reduction is performed or the output of the function kmeans for the clustering algorithm. |

## Author(s)

Markus Ruschhaupt mailto:m.ruschhaupt@dkfz.de

## See Also

MCRestimate

## Examples

```
library(MCRestimate)
m <- matrix(c(rnorm(10,2,0.5),rnorm(10,4,0.5),rnorm(10,7,0.5),rnorm(10,2,0.5),rnorm(10,4,
cluster.kmeans.mean(m ,number.clusters=3)
```

---

# important.variable.names

*Writing tables with variable information*

---

## Description

The information slot of a MCRestimate object may contain lists of variable names that are important for each classification. This function produces summary tables of these variables. It is assumed that the first column of each information list contains the variable names if not otherwise specified with the argument listName. This is important if someone wants to write a new wrapper for a classification method.

## Usage

```
important.variable.names(mcr,file="important_variables",listName=NULL,writeFile=
```

## Arguments

| | |
|---|---|
| mcr | an object of class MCRestimate |
| file | a character string specifies the name of the output files |
| listName | a character string specifying the variable names vector in the information list |
| writeFile | Should the files be written? |
| ... | Further arguments that are passed on to plot.default |

## Value

The function is called for its side effect, writing two tables with variable information.

## Author(s)

Markus Ruschhaupt mailto:m.ruschhaupt@dkfz.de

## Examples

```
library(MCRestimate)
library(golubEsets)
data(Golub_Train)


class.column <- "ALL.AML"
list.of.poss.parameter <- list(var.numbers = c(250,1000))
Preprocessingfunctions <- c("identity")
```

```
list.of.poss.parameter <- list(threshold = 6)
class.function <- "PAM.wrap"
plot.label <- "Samples"

cross.outer <- 10
cross.repeat <- 7
cross.inner <- 5

PAM.estimate <- MCRestimate(Golub_Train,
class.column,
classification.fun = class.function,
thePreprocessingMethods = Preprocessingfunctions,
poss.parameters = list.of.poss.parameter,
cross.outer = cross.outer,
cross.inner = cross.inner,
cross.repeat = cross.repeat,
plot.label = plot.label)

important.variable.names(PAM.estimate)
```

---

`MCRindError` *Individual Error of the outer cross-validations*

---

## Description

`MCRindError` returns a vector with the individual number of incorrect classified samples for each cross-validation `plotIndGroupVotes` plots the individual group votes

## Usage

```
MCRindError(MCRe,
perGroup=FALSE)

plotIndGroupVotes(MCRest,
PvD= 0.5,
dotCol="red",
errCol="black",
xlab="",
ylab="# misclassified samples (mean + SD)",
...)
```

## Arguments

| | |
|---|---|
| `MCRe` | Object of S3 class `MCRestimate` |
| `perGroup` | returns a vector with the individual number of incorrect classified samples for each group |
| `MCRest` | Object of S3 class `MCRestimate` |
| `PvD` | Offset of the text that belongs to a specific point in the plot |
| `dotCol` | Color of the dots |
| `errCol` | Line color between points |
| `xlab` | Label of X-Axis |

| | |
|---|---|
| ylab | Label of Y-Axis |
| ... | Advanced options to the plot command |

### Value

MCRindError returns a vector of individual errors.

### Author(s)

Markus Ruschhaupt mailto:m.ruschhaupt@dkfz.de

### See Also

MCRestimate

### Examples

```
library(MCRestimate)
library(golubEsets)
data(Golub_Train)
exSet <- Golub_Train[1:500,]
result1 <-MCRestimate(exSet,"ALL.AML",classification.fun="RF.wrap",cross.outer=3,cross.re
MCRindError(result1)
```

---

## intersectList *A function for creating a all possible intersects for a list of sets.*

---

### Description

The list contains several sets. The function calculates all possible intersections.

### Usage

```
intersectList(x)
```

### Arguments

| | |
|---|---|
| x | a list |

### Value

A list containing all possible intersections.

### Author(s)

Markus Ruschhaupt mailto:m.ruschhaupt@dkfz.de

### Examples

```
library(MCRestimate)
a <- list (x=1:5,y=3:4,z=c(1,3,3))
intersectList(a)
```

MCRconfusion *Summary tables for MCRestimate objects*

## Description

MCRwrongsamples returns a matrix with all the samples that have a higher frequency of being predicted as a member of a wrong class than of the correct class for at least one classification method. MCRconfusion summarizes the result of the vote matices

## Usage

```
MCRwrongsamples(x,
          col.names=names(x),
          rownames.from.object=TRUE,
          subgroup=NULL,
          freq=FALSE)

MCRconfusion(x,
              col.names=names(x),
              row.names=NULL)
```

## Arguments

| | |
|---|---|
| x | List of objects of S3 class MCRestimate |
| col.names | Vector of strings used for column names. The length must match the number of objects in x |
| rownames.from.object | Logical. If TRUE then the sample names of the MCRestimate object in x are used as row names |
| subgroup | Logical. If TRUE then only the samples which belongs to the specified group are listed in the table |
| freq | Logical. If TRUE then the frequency with which each sample in the table has been misclassified will be printed. |
| row.names | Vector of strings used for row names. If not specified the names of the groups are used |

## Value

MCRwrongsamples returns a matrix and MCRconfusion returns a confusion matrix.

## Author(s)

Markus Ruschhaupt mailto:m.ruschhaupt@dkfz.de

## See Also

MCRestimate

### Examples

```
library(MCRestimate)
library(golubEsets)
data(Golub_Train)
exSet <- Golub_Train[1:500,]
result1 <- MCRestimate(exSet,"ALL.AML",classification.fun="RF.wrap",cross.outer=3,cross.r
result2 <- MCRestimate(exSet,"ALL.AML",classification.fun="PAM.wrap",poss.parameters=list
MCRwrongsamples(list(result1,result2),subgroup="AML",col.names=c("Random Forest","PAM"))
MCRconfusion(list(result1,result2),col.names=c("Random Forest","PAM"))
```

---

## MCRestimateMerge *Merging objects of class MCRestimate*

---

### Description

MCRestimateMerge merges objects of S3 class `MCRestimate` See vignette MCRestimateBlueprint.Rnw for details.

### Usage

```
MCRestimateMerge(MCRestimateList)
```

### Arguments

`MCRestimateList` list of objects of S3 class `MCRestimate`

### Value

An objects of S3 class `MCRestimate`

### Author(s)

Markus Ruschhaupt mailto:m.ruschhaupt@dkfz.de

### See Also

MCRestimate

---

## MCRestimate *Estimation of misclassification error by cross-validation*

---

### Description

Several repetitions of a cross-validation are performed to get 'votes' how stable a method is against different partitions into training and test set

## Usage

```
MCRestimate(eset,
            class.column,
            reference.class=NULL,
            classification.fun,
            variableSel.fun="identity",
            cluster.fun="identity",
            poss.parameters=list(),
            cross.outer=10,
            cross.repeat=3,
            cross.inner=cross.outer,
            plot.label=NULL,
            rand=123,
            stratify=FALSE,
            information=TRUE,
            block.column=NULL,
            thePreprocessingMethods=c(variableSel.fun,cluster.fun))
```

## Arguments

| | |
|---|---|
| `eset` | an object of class `ExpressionSet` |
| `class.column` | a number or a character string which indicates the column of the expression set's phenodata containing the class label |
| `reference.class` | a character string - the name of one class - if specified, the class will form the first class and all the other classes will form the second class |
| `classification.fun` | character string which names the function that should be used for the classification |
| `variableSel.fun` | character string which names the function that should be used for the variable selection |
| `cluster.fun` | character string which names the function that should be used for the clustering of variables |
| `thePreprocessingMethods` | vector of character with the names of all preprocessing functions - can be used instead of 'variableSel.fun' and 'cluster.fun' - see details |
| `poss.parameters` | a list of possible values for the parameter of the classification, variable selection, and cluster methods |
| `cross.outer` | integer - the number of nearly equal sized parts the sample set should be divided into (outer cross-validation) |
| `cross.repeat` | integer - the number of repetitions of the cross-validation procedure |
| `cross.inner` | integer - the number of nearly equal sized parts the train set should be divided into (inner cross-validation) |
| `plot.label` | name of one column of the phenodata- if specified, the content of this column will form the labels of the x-axis if the 'votematrix' will be plotted with plot.MCRestimate |
| `rand` | integer - the random number generator will be put in a reproducible state |
| `stratify` | should a stratified version be used for the cross validation? |

`block.column` a character string which indicates the column of the expression set's phenodata containing the blocking covariate, which sets a constrain on the cross-validation splits: each block is either completely assigned to the test or to the training set

`information` information - should classificator specific data be given(depends on the wrapper for the classification method)

## Details

The argument 'thePreprocessingMethods' can be used instead of 'variableSel.fun' and 'cluster.fun'. In the first versions of MCRestimate it was only possible to have one variable selection and one cluster functions. Now it is possible to have more than two functions and the ordering is arbitrary, e.g. you can have a variable selection function, then a cluster function and then a seccond variable selection function.

If MCRestimate is used with an object of class `exprSetRG-class`, the preprocessing steps can use the green and the red channel seperately but the classification methods works with green channel - red channel.

Note: 'correct prediction' means that a sample was predicted to be a member of the correct class at least as often as it was predicted to be a member of each other class. So in the two class problem a sample is also 'correct' if it has been predicted correctly half of the time.

## Value

an object of class `MCRestimate` which is a `list` with fourteen arguments:

`votes` a matrix consisting of the different votes for each sample

`classes` the class of each sample

`table` a 'confusion' table, shows the number of 'correct prediction' for each class

`correct.prediction` a logical vector - indicates if a sample was predicted to be a member of the correct class at least as often as it was predicted to be a member of each other class.

`correct.class.vote` vector that contains for every sample the vote for it's correct class

`parameter` a list consisting of the estimated 'best' parameter for each cross-validation part

`class.method` string which names the function used for the classification

`thePreprocessingMethods` character string - name of the preprocessing functions that have been used

`cross.outer` number of blocks for a the outer cross-validation

`cross.repeat` number of outer cross-validation repetitions

`cross.inner` number of blocks for a the inner cross-validation

`sample.names` names of the sample

`information` classificator specific data (if information is TRUE)

## Author(s)

Markus Ruschhaupt mailto:m.ruschhaupt@dkfz.de, contributions from Andreas Buness and Patrick Warnat

## Examples

```
library(MCRestimate)
library(golubEsets)
data(Golub_Test)
G2 <- Golub_Test[1:500,]
result <- MCRestimate(G2, "ALL.AML", classification.fun="RF.wrap",
                      cross.outer=4, cross.repeat=3)
result
if (interactive()) {
  x11(width=9, height=4)}
plot(result)
```

---

`plot.MCRestimate` *Plot method for a objects of class MCRestimate*

---

## Description

plot.MCRestimate visualizes a 'vote matrix'. A 'vote matrix' is the result of a classification procedure. For every sample (=row) i and every class (=column) j the matrix element [i,j] is the probability or frequency the classification method predicts sample i as a member of class j.

## Usage

```
## S3 method for class 'MCRestimate':
plot(x,
                  class.factor=NULL,
                  rownames.from.object=FALSE,
                  sample.order=TRUE,
                  legend=FALSE,
                  mypalette=NULL,
                  shading=NULL,
                  xlab="Sample ID",
                  ylab="Frequency of correct classification",
                  cex.axis=1,...)
```

## Arguments

| | |
|---|---|
| `x` | Object of S3 class `MCRestimate` or a matrix |
| `class.factor` | Factor. Its length must match the number of rows in `x` and the levels must be the same as the colnames in `x`. If `x` is of class `MCRestimate` this argument will be ignored. |
| `rownames.from.object` | Logical. If TRUE then the rownames of the matrix or the sample names of `MCRestimate` in `x` are used as labels for the x-axis |
| `sample.order` | Logical. If TRUE then the samples are ordered by class membership |
| `legend` | Logical. If TRUE then there will be a small legend in the output |
| `mypalette` | vector with length equal to the number of classes. The vector specifies the color for the bar representing the classes. If 'NULL' colors choosen by the author are used. |

| | |
|---|---|
| shading | the density of shading lines for the rectagles that indicate the groups, in lines per inch. The default value of 'NULL' means that no shading lines are drawn. |
| xlab | Character |
| ylab | Character |
| cex.axis | numeric |
| ... | Further arguments that are passed on to plot.default |

## Value

The function is called for its side effect, creating a plot on the active graphics device.

## Author(s)

Markus Ruschhaupt mailto:m.ruschhaupt@dkfz.de

## See Also

MCRestimate

## Examples

```
  library(MCRestimate)
  x <- c(0.5, 0.3, 0.7, 0.3, 0.8, 0.2, 0.3)
  mat2 <- cbind(x, 1-x)
  fac2 <- factor(c("low", rep("high", 3), rep("low", 3)))
  colnames(mat2) <- levels(fac2)

  mat3 <- cbind(x/3, 2*x/3, 1-x)
  fac3 <- factor(c(rep("high", 3), rep("intermediate", 2), rep("low", 2)))
  colnames(mat3) <- levels(fac3)
if (interactive()) {
  x11(width=9, height=9)
  par(mfrow=c(3,1))}
  plot.MCRestimate(mat2, fac2)
  plot.MCRestimate(mat2, fac2, sample.order=FALSE)
  plot.MCRestimate(mat3, fac3)
```

---

# PLR *A function which performs penalised logistic regression classification for two groups*

---

## Description

A function which performs penalised logistic regression.

## Usage

```
PLR(trainmatrix, resultvector, kappa=0, eps=1e-4)
       ## S3 method for class 'PLR':
predict(object,...)
```

### Arguments

| | |
|---|---|
| resultvector | a vector which contains the labeling of the samples |
| trainmatrix | a matrix which includes the data. The rows corresponds to the observations and the colums to the variables. |
| kappa | value range for penalty parameter. If more that one parameter is specified the one with the lowest AIC will be used. |
| eps | precision of convergence |
| object | a fitted PLR model |
| ... | here a data matrix from samples that should be predicted |

### Value

a list with three arguments

| | |
|---|---|
| a | Intercept estimate of the linear predictor |
| b | vector of estimated regresion coefficients |
| factorlevel | levels of grouping variable |
| aics | vector of AIC values with respect to penalty parameter kappa |
| trs | vector of effective degrees of freedom with respect to penalty parameter kappa |

### Author(s)

Axel Benner, Ulrich Mansmann, based on MathLab code by Paul Eilers

### Examples

```
library(golubEsets)
data(Golub_Merge)
eSet<-Golub_Merge
X0 <- t(exprs(eSet))
m <- nrow(X0); n <- ncol(X0)
y <- pData(eSet)$ALL.AML
f <- PLR(X0, y,kappa=10^seq(0, 7, 0.5))
if (interactive()) {
  x11(width=9, height=4)
  par(mfrow=c(1,2))
plot(log10(f$kappas), f$aics, type="l",main="Akaike's Information Criterion", xlab="log k
plot(log10(f$kappas), f$trs, type="l",xlab="log kappa",
ylab="Dim",main="Effective dimension")
}
```

---

## replace.NA *Replaces in a given numeric matrix NA values per row or per column.*

---

### Description

Replaces in a given numeric matrix NA values per row or per column.

### Usage

```
replace.NA(x, replacement, byRow = TRUE)
```

### Arguments

| | |
|---|---|
| x | numeric input matrix |
| replacement | numeric vector containing the values which are used for NA replacement. If byRow = TRUE, this vector must contain as many values as matrix X has rows. Else, this vector must contain as many elements as matrix X has columns. |
| byRow | logical. If TRUE, then NA values in row n are replaced by the value at position n in the vector 'replacement'. Else, NA values are replaced according to their column position. |

### Value

The numeric input matrix with replaced NA values.

### Author(s)

Patrick Warnat mailto:p.warnat@dkfz-heidelberg.de

---

## select.NA.elements *Selects NA values of a given numeric matrix*

---

### Description

Selects of a given numeric matrix rows or columns containing more NA values than defined by a given threshold.

### Usage

```
select.NA.elements(x, NAthreshold, byRow = TRUE)
```

### Arguments

| | |
|---|---|
| x | numeric input matrix |
| NAthreshold | numeric value between 0 and 1, determining the allowed percentage of NA values per row or column. Rows or columns containing more NA values are selected. |
| byRow | logical. If TRUE, then rows with more NA values than determined by the argument threshold are selected, else columns. |

### Value

logical vector containing the row or column selection. If argument byRow = TRUE, then value contains as many values as the input matrix contains rows, else it contains as many values as the input matrix contains columns.

### Author(s)

Patrick Warnat mailto:p.warnat@dkfz-heidelberg.de

SVM.OVA.wrap *SVM with 'One-Versus-All' multiclass approach*

## Description

Multiclass approach where k binary SVM classifiers are constructed for a classification problem with k classes: Every classifier is trained to distinguish samples of one class from samples of all other classes. For prediction of the class of a new sample, the sample is classified by all k classifiers, and the class corresponding to the classifier with the maximum decision value is chosen.

## Usage

```
SVM.OVA.wrap(x,y,gamma = NULL, kernel = "radial", ...)
```

## Arguments

| | |
|---|---|
| x,y | x is a matrix where each row refers to a sample and each colum refers to a gene; y is a factor which includes the class for each sample |
| gamma | parameter for support vector machines |
| kernel | parameter for support vector machines |
| ... | Further parameters |

## Value

A predict function which can be used to predict the classes for a new data set.

## Author(s)

Patrick Warnat mailto:p.warnat@dkfz-heidelberg.de

## See Also

MCRestimate

## Examples

```
## Not run:
library(MCRestimate)
library(golubEsets)
data(Golub_Train)

class.column <- "ALL.AML"
Preprocessingfunctions <- c("varSel.highest.var")
list.of.poss.parameter <- list(var.numbers = c(250,1000))

Preprocessingfunctions <- c("identity")
class.function <- "SVM.OVA.wrap"
list.of.poss.parameter <- list(gamma = 6)
plot.label <- "Samples"

cross.outer <- 10
cross.repeat <- 20
```

```
cross.inner <- 5

SVM.estimate <- MCRestimate(Golub_Train,
class.column,
classification.fun = class.function,
thePreprocessingMethods = Preprocessingfunctions,
poss.parameters = list.of.poss.parameter,
cross.outer = cross.outer, cross.inner = cross.inner,
cross.repeat = cross.repeat, plot.label = plot.label)


## End(Not run)
```

---

`RF.wrap` *Wrapper function for different classification methods*

---

## Description

Wrapper function for different classification methods used by `MCRestimator`. These functions are mainly used within the function `MCRestimate`

## Usage

```
RF.wrap(x,y,...)
PAM.wrap(x,y,threshold,...)
PLR.wrap(x,y,kappa=0,eps=1e-4,...)
SVM.wrap(x,y,gamma = NULL, kernel = "radial", ...)
GPLS.wrap(x,y,...)
```

## Arguments

| | |
|---|---|
| `x,y` | x is a matrix where each row refers to a sample a each colum refers to a gene; y is a factor which includes the class for each sample |
| `threshold` | the threshold for PAM |
| `kappa` | the penalty parameter for the penalised logistic regression |
| `eps` | |
| `gamma` | parameter for support vector machines |
| `kernel` | parameter for support vector machines |
| `...` | Further parameters |

## Value

Every function return a predict function which can be used to predict the classes for a new data set.

## Author(s)

Markus Ruschhaupt `mailto:m.ruschhaupt@dkfz.de`

## See Also

`MCRestimate`

## Examples

```
library(MCRestimate)
library(golubEsets)
data(Golub_Train)

class.column <- "ALL.AML"
Preprocessingfunctions <- c("varSel.highest.var")
list.of.poss.parameter <- list(threshold = 6)

Preprocessingfunctions <- c("identity")
class.function <- "PAM.wrap"
plot.label <- "Samples"

cross.outer <- 10
cross.repeat <- 7
cross.inner <- 5

PAM.estimate <- MCRestimate(Golub_Train,
class.column,
classification.fun = class.function,
thePreprocessingMethods = Preprocessingfunctions,
poss.parameters = list.of.poss.parameter,
cross.outer = cross.outer, cross.inner = cross.inner,
cross.repeat = cross.repeat, plot.label = plot.label)
```

# Index