

# Package ‘pRoloc’

October 9, 2013

**Type** Package

**Title** A unifying bioinformatics framework for spatial proteomics

**Version** 1.0.1

**Author@R** c(person(given = ‘Laurent’, family = ‘Gatto’, email = ‘lg390@cam.ac.uk’, role = c(‘aut’, ‘cre’)), person(given = ‘Lisa’, family = ‘Breckels’, email = ‘lms79@cam.ac.uk’, role = ‘aut’), person(given = ‘Samuel’, family = ‘Wieczorek’, email = ‘samuel.wieczorek@cea.fr’, role = ‘ctb’))

**Author** Laurent Gatto and Lisa M. Breckels with contributions from Thomas Burger and Samuel Wieczorek

**Maintainer** Laurent Gatto <lg390@cam.ac.uk>

**Description** This package implements pattern recognition techniques on quantitative mass spectrometry data to infer protein sub-cellular localisation.

**Depends** R (>= 2.15), MSnbase (>= 1.7.23), MLInterfaces (>= 1.37.1), methods

**Imports** mclust, MSBVAR, caret, e1071, sampling, class, kernlab, lattice, nnet, randomForest, proxy, BiocGenerics, stats4, RColorBrewer, scales, MASS, knitr

**Suggests** testthat, pRolocdata, roxygen2

**License** GPL-2

**VignetteBuilder** knitr

## R topics documented:

addLegend . . . . .	2
addMarkers . . . . .	3
chi2-methods . . . . .	4
empPvalues . . . . .	5
exprsToRatios-methods . . . . .	6
GenRegRes-class . . . . .	6

getMarkers . . . . .	8
getPredictions . . . . .	8
getStockcol . . . . .	9
knnClassification . . . . .	11
knnOptimisation . . . . .	12
ksvmClassification . . . . .	13
ksvmOptimisation . . . . .	14
makeNaData . . . . .	15
minClassScore . . . . .	16
minMarkers . . . . .	17
MLearn-methods . . . . .	18
nbClassification . . . . .	19
nbOptimisation . . . . .	20
nnetClassification . . . . .	21
nnetOptimisation . . . . .	22
perTurboClassification . . . . .	23
perTurboOptimisation . . . . .	24
phenoDisco . . . . .	25
plot2D . . . . .	27
plotDist . . . . .	28
plsdaClassification . . . . .	29
plsdaOptimisation . . . . .	31
rfClassification . . . . .	32
rfOptimisation . . . . .	33
svmClassification . . . . .	34
svmOptimisation . . . . .	35
<b>Index</b>	<b>36</b>

---

addLegend	<i>Adds a legend</i>
-----------	----------------------

---

## Description

Adds a legend to a [plot2D](#) figure.

## Usage

```
addLegend(object, fcol = "markers", where = "other", col,
          ...)
```

## Arguments

object	An instance of class MSnSet
fcol	Feature meta-data label (fData column name) defining the groups to be differentiated using different colours. Default is markers.

where	One of "other", "bottomleft", "bottomright", "topleft" or "topright" defining the location of the legend. "other" opens a new graphics device, while the other locations are passed to <a href="#">legend</a> .
col	A character defining point colours.
...	Additional parameters passed to <a href="#">legend</a> .

**Value**

Invisibly returns NULL

**Author(s)**

Laurent Gatto

---

addMarkers	<i>Adds markers to the data</i>
------------	---------------------------------

---

**Description**

The function adds a 'markers' feature variable. These markers are read from a comma separated values (csv) spreadsheet file. This markers file is expected to have 2 columns (others are ignored) where the first is the name of the marker features and the second the group label. It is essential to assure that featureNames(object) and marker names (first column) match, i.e. the same feature identifiers and case fold are used.

**Usage**

```
addMarkers(object, markerfile, verbose = TRUE)
```

**Arguments**

object	An instance of class MSnSet.
markerfile	A character with the name the markers' csv file.
verbose	A logical indicating if number of markers and marker table should be printed to the console.

**Value**

A new instance of class MSnSet with an additional markers feature variable.

**Author(s)**

Laurent Gatto

## Description

In the original protein correlation profiling (PCP), Andersen et al. use the peptide normalised profiles along gradient fractions and compared them with the reference profiles (or set of profiles) by computing  $Chi^2$  values,  $\frac{\sum(x_i - x_p)^2}{x_p}$ , where  $x_i$  is the normalised value of the peptide in fraction  $i$  and  $x_p$  is the value of the marker (from Wiese et al., 2007). The protein  $Chi^2$  is then computed as the median of the peptide  $Chi^2$  values. Peptides and proteins with similar profiles to the markers will have small  $Chi^2$  values.

The chi2 methods implement this idea and compute such  $Chi^2$  values for sets of proteins.

## Methods

`signature(x = "matrix", y = "matrix", method = "character", fun = "NULL", na.rm = "logical")`

Compute `nrow(x)` times `nrow(y)`  $Chi^2$  values, for each  $x, y$  feature pair. Method is one of "Andersen2003" or "Wiese2007"; the former (default) computed the  $Chi^2$  as  $\sum(y-x)^2/\text{length}(x)$ , while the latter uses  $\sum((y-x)^2/x)$ . `na.rm` defines if missing values (NA and NaN) should be removed prior to summation. `fun` defines how to summarise the  $Chi^2$  values; default, NULL, does not combine the  $Chi^2$  values.

`signature(x = "matrix", y = "numeric", method = "character", na.rm = "logical")`

Computes `nrow(x)`  $Chi^2$  values, for all the  $(x_i, y)$  pairs. See above for the other arguments.

`signature(x = "numeric", y = "matrix", method = "character", na.rm = "logical")`

Computes `nrow(y)`  $Chi^2$  values, for all the  $(x, y_i)$  pairs. See above for the other arguments.

`signature(x = "numeric", y = "numeric", method = "character", na.rm = "logical")`

Computes the  $Chi^2$  value for the  $(x, y)$  pairs. See above for the other arguments.

## Author(s)

Laurent Gatto <lg390@cam.ac.uk>

## References

Andersen, J. S., Wilkinson, C. J., Mayor, T., Mortensen, P. et al., Proteomic characterization of the human centrosome by protein correlation profiling. *Nature* 2003, 426, 570 - 574.

Wiese, S., Gronemeyer, T., Ofman, R., Kunze, M. et al., Proteomics characterization of mouse kidney peroxisomes by tandem mass spectrometry and protein correlation profiling. *Mol. Cell. Proteomics* 2007, 6, 2045 - 2057.

## See Also

[empPvalues](#)

## Examples

```
mrk <- rnorm(6)
prot <- matrix(rnorm(60), ncol = 6)
chi2(mrk, prot, method = "Andersen2003")
chi2(mrk, prot, method = "Wiese2007")

pepmark <- matrix(rnorm(18), ncol = 6)
pepprot <- matrix(rnorm(60), ncol = 6)
chi2(pepmark, pepprot)
chi2(pepmark, pepprot, fun = sum)
```

---

empPvalues

*Estimate empirical p-values for  $Chi^2$  protein correlations.*

---

## Description

Andersen et al. (2003) used a fixed  $Chi^2$  threshold of 0.05 to identify organelle-specific candidates. This function computes empirical p-values by permutation the markers relative intensities and computed null  $Chi^2$  values.

## Usage

```
empPvalues(marker, corMatrix, n = 100, ...)
```

## Arguments

marker	A numerics with markers relative intensities.
corMatrix	A matrix of <code>nrow(corMatrix)</code> protein relative intensities to be compares against the marker.
n	The number of iterations.
...	Additional parameters to be passed to <code>chi2</code> .

## Value

A numeric of length `nrow(corMatrix)`.

## Author(s)

Laurent Gatto <lg390@cam.ac.uk>

## References

Andersen, J. S., Wilkinson, C. J., Mayor, T., Mortensen, P. et al., Proteomic characterization of the human centrosome by protein correlation profiling. Nature 2003, 426, 570 - 574.

## See Also

[chi2](#) for  $Chi^2$  calculation.

**Examples**

```

set.seed(1)
mrk <- rnorm(6, 5, 1)
prot <- rbind(matrix(rnorm(120, 5, 1), ncol = 6),
              mrk + rnorm(6))
mrk <- mrk/sum(mrk)
prot <- prot/rowSums(prot)
empPvalues(mrk, prot)

```

---

exprsToRatios-methods *Calculate all ratio pairs*

---

**Description**

Calculations all possible ratios for the assayData columns in an "MSnSet".

**Methods**

signature(object = "MSnSet", log = "logical") If log is FALSE (default) the ratios for all the assayData columns are computed; otherwise, log ratios (differences) are calculated.

**Examples**

```

library("pRolocdata")
data(dunkley2006)
x <- dunkley2006[, 1:3]
head(exprs(x))
r <- exprsToRatios(x)
head(exprs(r))
pData(r)

```

---

GenRegRes-class *Class "GenRegRes"*

---

**Description**

Regularisation framework container.

**Objects from the Class**

Object of this class are created with the respective regularisation function: [knnRegularisation](#), [svmRegularisation](#), [plsdaRegularisation](#), ...

**Slots**

**algorithm:** Object of class "character" storing the machine learning algorithm name.

**hyperparameters:** Object of class "list" with the respective algorithm hyper-parameters tested.

**design:** Object of class "numeric" describing the cross-validation design, the test data size and the number of replications.

**log:** Object of class "list" with warnings thrown during the hyper-parameters regularisation.

**seed:** Object of class "integer" with the random number generation seed.

**results:** Object of class "matrix" of dimensions times (see design) by number of hyperparameters + 1 storing the macro F1 values for the respective best hyper-parameters for each replication.

**f1Matrices:** Object of class "list" with respective times cross-validation F1 matrices.

**cmMatrices:** Object of class "list" with respective times contingency matrices.

**testPartitions:** Object of class "list" with respective times test partitions.

**datasize:** Object of class "list" with details about the respective inner and outer training and testing data sizes.

**Methods**

**getF1Scores** signature(object = "GenRegRes"): ...

**f1Count** signature(object = "GenRegRes", t = "numeric"): Constructs a table of all possible parameter combination and count how many have an F1 scores greater or equal than t. When t is missing (default), the best F1 score is used. This method is useful in conjunctin with plot.

**getRegularisedParams** signature(object = "GenRegRes"): ...

**getRegularizedParams** signature(object = "GenRegRes"): ...

**getSeed** signature(object = "GenRegRes"): ...

**getWarnings** signature(object = "GenRegRes"): ...

**levelPlot** signature(object = "GenRegRes"): ...

**plot** signature(x = "GenRegRes", y = "missing"): ...

**show** signature(object = "GenRegRes"): ...

**Author(s)**

Laurent Gatto <lg390@cam.ac.uk>

**Examples**

```
showClass("GenRegRes")
```

---

getMarkers	<i>Returns the organelle markers in an 'MSnSet'</i>
------------	---

---

### Description

Convenience accessor to the organelle markers in an 'MSnSet'. This function returns the organelle markers of an MSnSet instance. As a side effect, it print out a marker table.

### Usage

```
getMarkers(object, fcol = "markers", verbose = TRUE)
```

### Arguments

object	An instance of class "MSnSet".
fcol	The name of the markers column in the featureData slot. Default is markers.
verbose	If TRUE, a marker table is printed and the markers are returned invisibly. If FALSE, the markers are returned.

### Value

A character of length ncol(object).

### Author(s)

Laurent Gatto

### Examples

```
library("pRolocdata")
data(dunkley2006)
mymarkers <- getMarkers(dunkley2006)
```

---

getPredictions	<i>Returns the predictions in an 'MSnSet'</i>
----------------	---

---

### Description

Convenience accessor to the predicted feature localisation in an 'MSnSet'. This function returns the predictions of an MSnSet instance. As a side effect, it prints out a prediction table.

### Usage

```
getPredictions(object, fcol, scol, t = 0, verbose = TRUE)
```



**Arguments**

object	An instance of class "MSnSet".
fcol	The name of the prediction column in the featureData slot.
scol	The name of the prediction score column in the featureData slot. If missing, created by pasting '.scores' after fcol. If NULL, ignored.
t	The score threshold. Predictions with score < t are set to 'unknown'. Default is 0.
verbose	If TRUE, a prediction table is printed and the predictions are returned invisibly. If FALSE, the predictions are returned.

**Value**

A character of length ncol(object).

**Author(s)**

Laurent Gatto

---

getStockcol

*Manage default colours and point characters*

---

**Description**

These functions allow to get/set the default colours and point character that are used when plotting organelle clusters and unknown features. These values are parametrised at the session level.

**Usage**

```
getStockcol()
```

```
setStockcol(cols)
```

```
getStockpch()
```

```
setStockpch(pchs)
```

```
getUnknowncol()
```

```
setUnknowncol(col)
```

```
getUnknownpch()
```

```
setUnknownpch(pch)
```

**Arguments**

cols	A vector of colour characters or NULL, which sets the colours to the default values.
pchs	A vector of numeric or NULL, which sets the point characters to the default values.
col	A colour character or NULL, which sets the colour to #E7E7E7 (grey91), the default colour for unknown features.
pch	A numeric vector of length 1 or NULL, which sets the point character to 21, the default.

**Value**

A character vector.

Invisibly returns cols.

A numeric vector.

Invisibly returns pchs.

A character vector or length 1.

Invisibly returns col.

A numeric vector of length 1.

Invisibly returns pch.

**Author(s)**

Laurent Gatto

**Examples**

```
## defaults for clusters
getStockcol()
getStockpch()
## unknown features
getUnknownpch()
getUnknowncol()
## an example
library(pRolocdata)
data(dunkley2006)
par(mfrow = c(2, 1))
plot2D(dunkley2006, fcol = "markers", main = 'Default colours')
setUnknowncol("black")
plot2D(dunkley2006, fcol = "markers", main = 'setUnknowncol("black")')
getUnknowncol()
setUnknowncol(NULL)
getUnknowncol()
```

---

knnClassification      *knn classification*

---

## Description

Classification using for the k-nearest neighbours algorithm.

## Usage

```
knnClassification(object, assessRes,  
  scores = c("prediction", "all", "none"), k,  
  fcol = "markers", ...)
```

## Arguments

object	An instance of class "MSnSet".
assessRes	An instance of class "GenRegRes", as generated by <a href="#">knnOptimisation</a> .
scores	One of "prediction", "all" or "none" to report the score for the predicted class only, for all cluster or none.
k	If assessRes is missing, a k must be provided.
fcol	The feature meta-data containing marker definitions. Default is markers.
...	Additional parameters passed to <a href="#">knn</a> from package <code>class</code> .

## Value

An instance of class "MSnSet" with `knn` and `knn.scores` feature variables storing the classification results and scores respectively.

## Author(s)

Laurent Gatto

## Examples

```
library(pRolocdata)  
data(dunkley2006)  
## reducing parameter search space and iterations  
params <- knnOptimisation(dunkley2006, k = c(3, 10), times = 3)  
params  
plot(params)  
f1Count(params)  
levelPlot(params)  
getParams(params)  
res <- knnClassification(dunkley2006, params)  
getPredictions(res, fcol = "knn")  
getPredictions(res, fcol = "knn", t = 0.75)  
plot2D(res, fcol = "knn")
```

---

knnOptimisation      *knn parameter optimisation*

---

### Description

Classification parameter optimisation for the k-nearest neighbours algorithm.

### Usage

```
knnOptimisation(object, fcol = "markers", k = 3:12,  
  times = 100, test.size = 0.2, xval = 5, fun = mean,  
  seed, verbose = TRUE, ...)
```

### Arguments

object	An instance of class " <a href="#">MSnSet</a> ".
fcol	The feature meta-data containing marker definitions. Default is markers.
k	The hyper-parameter. Default values are 3:12.
times	The number of times internal cross-validation is performed. Default is 100.
test.size	The size of test data. Default is 0.2 (20 percent).
xval	The n-cross validation. Default is 5.
fun	The function used to summarise the xval macro F1 matrices.
seed	The optional random number generator seed.
verbose	A logical defining whether a progress bar is displayed.
...	Additional parameters passed to <a href="#">knn</a> from package class.

### Value

An instance of class "[GenRegRes](#)".

### Author(s)

Laurent Gatto

### See Also

[knnClassification](#) and example therein.

---

ksvmClassification      *ksvm classification*

---

## Description

Classification using the support vector machine algorithm.

## Usage

```
ksvmClassification(object, assessRes,
  scores = c("prediction", "all", "none"), cost,
  fcol = "markers", ...)
```

## Arguments

object	An instance of class "MSnSet".
assessRes	An instance of class "GenRegRes", as generated by <a href="#">ksvmOptimisation</a> .
scores	One of "prediction", "all" or "none" to report the score for the predicted class only, for all cluster or none.
cost	If assessRes is missing, a cost must be provided.
fcol	The feature meta-data containing marker definitions. Default is markers.
...	Additional parameters passed to <a href="#">ksvm</a> from package kernlab.

## Value

An instance of class "MSnSet" with ksvm and ksvm.scores feature variables storing the classification results and scores respectively.

## Author(s)

Laurent Gatto

## Examples

```
library(pRocdata)
data(dunkley2006)
## reducing parameter search space and iterations
params <- ksvmOptimisation(dunkley2006, cost = 2^seq(-1,4,5), times = 3)
params
plot(params)
f1Count(params)
levelPlot(params)
getParam(params)
res <- ksvmClassification(dunkley2006, params)
getPredictions(res, fcol = "ksvm")
getPredictions(res, fcol = "ksvm", t = 0.75)
plot2D(res, fcol = "ksvm")
```

---

ksvmOptimisation      *ksvm parameter optimisation*

---

### Description

Classification parameter optimisation for the support vector machine algorithm.

### Usage

```
ksvmOptimisation(object, fcol = "markers",
  cost = 2^(-4:4), times = 100, test.size = 0.2,
  xval = 5, fun = mean, seed, verbose = TRUE, ...)
```

### Arguments

object	An instance of class " <a href="#">MSnSet</a> ".
fcol	The feature meta-data containing marker definitions. Default is markers.
cost	The hyper-parameter. Default values are $2^{-4:4}$ .
times	The number of times internal cross-validation is performed. Default is 100.
test.size	The size of test data. Default is 0.2 (20 percent).
xval	The n-cross validation. Default is 5.
fun	The function used to summarise the xval macro F1 matrices.
seed	The optional random number generator seed.
verbose	A logical defining whether a progress bar is displayed.
...	Additional parameters passed to <a href="#">ksvm</a> from package kernlab.

### Value

An instance of class "[GenRegRes](#)".

### Author(s)

Laurent Gatto

### See Also

[ksvmClassification](#) and example therein.

---

makeNaData	<i>Create a data with missing values</i>
------------	--

---

**Description**

These functions take an instance of class "MSnSet" and sets randomly selected values to NA.

**Usage**

```
makeNaData(object, nNA, pNA, exclude)

makeNaData2(object, nRows, nNAs, exclude)

whichNA(x)
```

**Arguments**

object	An instance of class MSnSet.
nNA	The absolute number of missing values to be assigned.
pNA	The proportion of missing values to be assigned.
exclude	A vector to be used to subset object, defining rows that should not be used to set NAs.
nRows	The number of rows for each set.
nNAs	The number of missing values for each set.
x	A matrix or an instance of class MSnSet.

**Details**

makeNaData randomly selects a number nNA (or a proportion pNA) of cells in the expression matrix to be set to NA.

makeNaData2 will select length(nRows) sets of rows from object, each with nRows[i] rows respectively. The first set will be assigned nNAs[1] missing values, the second nNAs[2], ... As opposed to makeNaData, this permits to control the number of NAs per rows.

The whichNA can be used to extract the indices of the missing values, as illustrated in the example.

**Value**

An instance of class MSnSet, as object, but with the appropriate number/proportion of missing values. The returned object has an additional feature meta-data columns, nNA

**Author(s)**

Laurent Gatto

**Examples**

```

## Example 1
library(pRolocdata)
data(dunkley2006)
sum(is.na(dunkley2006))
dunkleyNA <- makeNaData(dunkley2006, nNA = 150)
processingData(dunkleyNA)
sum(is.na(dunkleyNA))
table(fData(dunkleyNA)$nNA)
naIdx <- whichNA(dunkleyNA)
head(naIdx)
## Example 2
dunkleyNA <- makeNaData(dunkley2006, nNA = 150, exclude = 1:10)
processingData(dunkleyNA)
table(fData(dunkleyNA)$nNA[1:10])
table(fData(dunkleyNA)$nNA)
## Example 3
nr <- rep(10, 5)
na <- 1:5
x <- makeNaData2(dunkley2006[1:100, 1:5],
                 nRows = nr,
                 nNAs = na)
processingData(x)
(res <- table(fData(x)$nNA))
stopifnot(as.numeric(names(res)[-1]) == na)
stopifnot(res[-1] == nr)
## Example 2
nr2 <- c(5, 12, 11, 8)
na2 <- c(3, 8, 1, 4)
x2 <- makeNaData2(dunkley2006[1:100, 1:10],
                 nRows = nr2,
                 nNAs = na2)
processingData(x2)
(res2 <- table(fData(x2)$nNA))
stopifnot(as.numeric(names(res2)[-1]) == sort(na2))
stopifnot(res2[-1] == nr2[order(na2)])
## Example 5
nr3 <- c(5, 12, 11, 8)
na3 <- c(3, 8, 1, 3)
x3 <- makeNaData2(dunkley2006[1:100, 1:10],
                 nRows = nr3,
                 nNAs = na3)
processingData(x3)
(res3 <- table(fData(x3)$nNA))

```



**Description**

This functions updates the classification results in an "MSnSet" based on a prediction score threshold  $t$ . All features with a score  $< t$  are set to 'unknown'. Note that the original levels are preserved while 'unknown' is added.

**Usage**

```
minClassScore(object, fcol, scol, t = 0)
```

**Arguments**

object	An instance of class "MSnSet".
fcol	The name of the markers column in the featureData slot.
scol	The name of the prediction score column in the featureData slot. If missing, created by pasting '.scores' after fcol.
t	The score threshold. Predictions with score $< t$ are set to 'unknown'. Default is 0.

**Value**

The original object with a modified `fData(object)[, fcol]` feature variable.

**Author(s)**

Laurent Gatto

**Examples**

```
library(pRocdata)
data(dunkley2006)
## random scores
fData(dunkley2006)$assigned.scores <- runif(nrow(dunkley2006))
getPredictions(dunkley2006, fcol = "assigned")
getPredictions(dunkley2006, fcol = "assigned", t = 0.5)
x <- minClassScore(dunkley2006, fcol = "assigned", t = 0.5)
getPredictions(x, fcol = "assigned")
all.equal(getPredictions(dunkley2006, fcol = "assigned", t = 0.5),
          getPredictions(x, fcol = "assigned"))
```

---

minMarkers

*Creates a reduced marker variable*

---

**Description**

This function updates an MSnSet instances and sets markers class to unknown if there are less than  $n$  instances.

**Usage**

```
minMarkers(object, n = 10, fcol = "markers")
```

**Arguments**

object	An instance of class "MSnSet".
n	Minumum of marker instances per class.
fcol	The name of the markers column in the featureData slot. Default is markers.

**Value**

An instance of class "MSnSet" with a new feature variables, named after the original fcol variable and the n value.

**Author(s)**

Laurent Gatto

**Examples**

```
library(pRocdata)
data(dunkley2006)
d2 <- minMarkers(dunkley2006, 20)
getMarkers(dunkley2006)
getMarkers(d2, fcol = "markers20")
```

---

MLearn-methods

*The MLearn interface for machine learning*


---

**Description**

This method implements MLInterfaces' MLean method for instances of the class "MSnSet".

**Methods**

```
signature(formula = "formula", data = "MSnSet", .method = "learnerSchema", trainInd = "numeric")
```

The learning problem is stated with the formula and applies the .method schema on the MSnSet data input using the trainInd numeric indices as train data.

```
signature(formula = "formula", data = "MSnSet", .method = "learnerSchema", trainInd = "xvalSpec")
```

In this case, an instance of [xvalSpec](#) is used for cross-validation.

```
signature(formula = "formula", data = "MSnSet", .method = "clusteringSchema", trainInd = "missing")
```

Hierarchical (hclustI), k-means (kmeansI) and partitioning around medoids (pamI) clustering algorithms using MLInterface's MLearn interface.

**See Also**

The MLInterfaces package documentation, in particular [MLearn](#).

---

nbClassification	<i>nb classification</i>
------------------	--------------------------

---

## Description

Classification using the naive Bayes algorithm.

## Usage

```
nbClassification(object, assessRes,  
  scores = c("prediction", "all", "none"), laplace,  
  fcol = "markers", ...)
```

## Arguments

object	An instance of class " <a href="#">MSnSet</a> ".
assessRes	An instance of class " <a href="#">GenRegRes</a> ", as generated by <a href="#">nbOptimisation</a> .
scores	One of "prediction", "all" or "none" to report the score for the predicted class only, for all cluster or none.
laplace	If assessRes is missing, a laplace must be provided.
fcol	The feature meta-data containing marker definitions. Default is markers.
...	Additional parameters passed to <a href="#">naiveBayes</a> from package <a href="#">e1071</a> .

## Value

An instance of class "[MSnSet](#)" with nb and nb.scores feature variables storing the classification results and scores respectively.

## Author(s)

Laurent Gatto

## Examples

```
library(pRolocdata)  
data(dunkley2006)  
## reducing parameter search space and iterations  
params <- nbOptimisation(dunkley2006, laplace = c(0, 5), times = 3)  
params  
plot(params)  
f1Count(params)  
levelPlot(params)  
getParams(params)  
res <- nbClassification(dunkley2006, params)  
getPredictions(res, fcol = "naiveBayes")  
getPredictions(res, fcol = "naiveBayes", t = 1)  
plot2D(res, fcol = "naiveBayes")
```

---

`nbOptimisation`*nb paramter optimisation*

---

**Description**

Classification algorithm parameter for the naive Bayes algorithm.

**Usage**

```
nbOptimisation(object, fcol = "markers",  
               laplace = seq(0, 5, 0.5), times = 100, test.size = 0.2,  
               xval = 5, fun = mean, seed, verbose = TRUE, ...)
```

**Arguments**

<code>object</code>	An instance of class <code>"MSnSet"</code> .
<code>fcol</code>	The feature meta-data containing marker definitions. Default is <code>markers</code> .
<code>laplace</code>	The hyper-parameter. Default values are <code>seq(0, 5, 0.5)</code> .
<code>times</code>	The number of times internal cross-validation is performed. Default is 100.
<code>test.size</code>	The size of test data. Default is 0.2 (20 percent).
<code>xval</code>	The n-cross validation. Default is 5.
<code>fun</code>	The function used to summarise the <code>xval</code> macro F1 matrices.
<code>seed</code>	The optional random number generator seed.
<code>verbose</code>	A logical defining whether a progress bar is displayed.
<code>...</code>	Additional parameters passed to <code>naiveBayes</code> from package <code>e1071</code> .

**Value**

An instance of class `"GenRegRes"`.

**Author(s)**

Laurent Gatto

**See Also**

[nbClassification](#) and example therein.

---

nnetClassification     *nnet classification*

---

### Description

Classification using the artificial neural network algorithm.

### Usage

```
nnetClassification(object, assessRes,
  scores = c("prediction", "all", "none"), decay, size,
  fcol = "markers", ...)
```

### Arguments

object	An instance of class "MSnSet".
assessRes	An instance of class "GenRegRes", as generated by <a href="#">nnetOptimisation</a> .
scores	One of "prediction", "all" or "none" to report the score for the predicted class only, for all cluster or none.
decay	If assessRes is missing, a decay must be provided.
size	If assessRes is missing, a size must be provided.
fcol	The feature meta-data containing marker definitions. Default is markers.
...	Additional parameters passed to <a href="#">nnet</a> from package nnet.

### Value

An instance of class "MSnSet" with `nnet` and `nnet.scores` feature variables storing the classification results and scores respectively.

### Author(s)

Laurent Gatto

### Examples

```
library(pRolocdata)
data(dunkley2006)
## reducing parameter search space and iterations
params <- nnetOptimisation(dunkley2006, decay = 10^(c(-1, -5)), size = c(5, 10), times = 3)
params
plot(params)
f1Count(params)
levelPlot(params)
getParams(params)
res <- nnetClassification(dunkley2006, params)
getPredictions(res, fcol = "nnet")
getPredictions(res, fcol = "nnet", t = 0.75)
plot2D(res, fcol = "nnet")
```

---

nnetOptimisation      *nnet parameter optimisation*

---

### Description

Classification parameter optimisation for artificial neural network algorithm.

### Usage

```
nnetOptimisation(object, fcol = "markers",
  decay = c(0, 10^(-1:-5)), size = seq(1, 10, 2),
  times = 100, test.size = 0.2, xval = 5, fun = mean,
  seed, verbose = TRUE, ...)
```

### Arguments

object	An instance of class " <a href="#">MSnSet</a> ".
fcol	The feature meta-data containing marker definitions. Default is markers.
decay	The hyper-parameter. Default values are $c(0, 10^{-1:-5})$ .
size	The hyper-parameter. Default values are $seq(1, 10, 2)$ .
times	The number of times internal cross-validation is performed. Default is 100.
test.size	The size of test data. Default is 0.2 (20 percent).
xval	The n-cross validation. Default is 5.
fun	The function used to summarise the xval macro F1 matrices.
seed	The optional random number generator seed.
verbose	A logical defining whether a progress bar is displayed.
...	Additional parameters passed to <a href="#">nnet</a> from package nnet.

### Value

An instance of class "[GenRegRes](#)".

### Author(s)

Laurent Gatto

### See Also

[nnetClassification](#) and example therein.

---

perTurboClassification  
*perTurbo classification*

---

### Description

Classification using the PerTurbo algorithm.

### Usage

```
perTurboClassification(object, assessRes,  
  scores = c("prediction", "all", "none"), pRegul, sigma,  
  inv, reg, fcol = "markers")
```

### Arguments

object	An instance of class " <a href="#">MSnSet</a> ".
assessRes	An instance of class " <a href="#">GenRegRes</a> ", as generated by <a href="#">svmRegularisation</a> .
scores	One of "prediction", "all" or "none" to report the score for the predicted class only, for all cluster or none.
pRegul	If assessRes is missing, a pRegul must be provided. See <a href="#">perTurboOptimisation</a> for details.
sigma	If assessRes is missing, a sigma must be provided. See <a href="#">perTurboOptimisation</a> for details.
inv	The type of algorithm used to invert the matrix. Values are : "Inversion Cholesky" ( <a href="#">chol2inv</a> ), "Moore Penrose" ( <a href="#">ginv</a> ), "solve" ( <a href="#">solve</a> ), "svd" ( <a href="#">svd</a> ). Default value is "Inversion Cholesky".
reg	The type of regularisation of matrix. Values are "none", "trunc" or "tikhonov". Default value is "tikhonov".
fcol	The feature meta-data containing marker definitions. Default is markers.

### Value

An instance of class "[MSnSet](#)" with perTurbo and perTurbo.scores feature variables storing the classification results and scores respectively.

### Author(s)

Thomas Burger and Samuel Wiczorek

### References

N. Courty, T. Burger, J. Laurent. "PerTurbo: a new classification algorithm based on the spectrum perturbations of the Laplace-Beltrami operator", The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2011), D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part I, LNAI 6911, pp. 359 - 374, Athens, Greece, September 2011.

**Examples**

```

library(pRolocdata)
data(dunkley2006)
## reducing parameter search space
params <- perTurboOptimisation(dunkley2006,
                              pRegul = 2^seq(-2,2,2),
                              sigma = 10^seq(-1, 1, 1),
                              inv = "Inversion Cholesky",
                              reg ="tikhonov",
                              times = 3)

params
plot(params)
f1Count(params)
levelPlot(params)
getParams(params)
res <- perTurboClassification(dunkley2006, params)
getPredictions(res, fcol = "perTurbo")
getPredictions(res, fcol = "perTurbo", t = 0.75)
plot2D(res, fcol = "perTurbo")

```

---

perTurboOptimisation *PerTurbo parameter optimisation*

---

**Description**

Classification parameter optimisation for the PerTurbo algorithm

**Usage**

```

perTurboOptimisation(object, fcol = "markers",
                     pRegul = 10^(seq(from = -1, to = 0, by = 0.2)),
                     sigma = 10^(seq(from = -1, to = 1, by = 0.5)),
                     inv = c("Inversion Cholesky", "Moore Penrose", "solve", "svd"),
                     reg = c("tikhonov", "none", "trunc"), times = 1,
                     test.size = 0.2, xval = 5, fun = mean, seed,
                     verbose = TRUE)

```

**Arguments**

object	An instance of class <i>"MSnSet"</i> .
fcol	The feature meta-data containing marker definitions. Default is markers.
inv	The type of algorithm used to invert the matrix. Values are : "Inversion Cholesky" ( <a href="#">chol2inv</a> ), "Moore Penrose" ( <a href="#">ginv</a> ), "solve" ( <a href="#">solve</a> ), "svd" ( <a href="#">svd</a> ). Default value is "Inversion Cholesky".
reg	The type of regularisation of matrix. Values are "none", "trunc" or "tikhonov". Default value is "tikhonov".



pRegul	The hyper-parameter for the regularisation (values are in ]0,1]). If reg == "trunc", pRegul is for the percentage of eigen values in matrix. If reg == "tikhonov", then 'pRegul' is the parameter for the tikhonov regularisation. Available configurations are : "Inversion Cholesky" - ("tikhonov" / "none"), "Moore Penrose" - ("tikhonov" / "none"), "solve" - ("tikhonov" / "none"), "svd" - ("tikhonov" / "none" / "trunc").
sigma	The hyper-parameter.
times	The number of times internal cross-validation is performed. Default is 100.
test.size	The size of test data. Default is 0.2 (20 percent).
xval	The n-cross validation. Default is 5.
fun	The function used to summarise the times macro F1 matrices.
seed	The optional random number generator seed.
verbose	A logical defining whether a progress bar is displayed.

**Value**

An instance of class "[GenRegRes](#)".

**Author(s)**

Thomas Burger and Samuel Wieczorek

**See Also**

[perTurboClassification](#) and example therein.

---

phenoDisco

*Runs the phenoDisco algorithm.*

---

**Description**

phenoDisco is a semi-supervised iterative approach to detect new protein clusters.

**Usage**

```
phenoDisco(object, fcol = "markers", times = 100,  
           GS = 10, allIter = FALSE, p = 0.05, seed,  
           verbose = TRUE)
```

**Arguments**

object	An instance of class MSnSet.
fcol	A character indicating the organellar markers column name in feature meta-data. Default is markers.
times	Number of runs of tracking. Default is 100.
GS	Group size, i.e how many proteins make a group. Default is 10 (the minimum group size is 4).
allIter	logical, defining if predictions for all iterations should be saved. Default is FALSE.
p	Significance level for outlier detection. Default is 0.05.
seed	An optional numeric of length 1 specifying the random number generator seed to be used.
verbose	Logical, indicating if messages are to be printed out during execution of the algorithm.

**Details**

The algorithm performs a phenotype discovery analysis as described in Breckels et al. Using this approach one can identify putative subcellular groupings in organelle proteomics experiments for more comprehensive validation in an unbiased fashion. The method is based on the work of Yin et al. and used iterated rounds of Gaussian Mixture Modelling using the Expectation Maximisation algorithm combined with a non-parametric outlier detection test to identify new phenotype clusters.

One requires 2 or more classes to be labelled in the data and at a very minimum of 6 markers per class to run the algorithm. The function will check and remove feature with missing values using the `filterNA` method.

Important: Prior to version 1.1.2 the row order in the output was different from the row order in the input. This has now been fixed and row ordering is now the same in both input and output objects.

**Value**

An instance of class MSnSet containing the phenoDisco predictions.

**Author(s)**

Lisa M. Breckels <lms79@cam.ac.uk>

**References**

Yin Z, Zhou X, Bakal C, Li F, Sun Y, Perrimon N, Wong ST. Using iterative cluster merging with improved gap statistics to perform online phenotype discovery in the context of high-throughput RNAi screens. *BMC Bioinformatics*. 2008 Jun 5;9:264. PubMed PMID: 18534020; PubMed Central PMCID: PMC2443381.

Breckels LM, Gatto L, Christoforou A, Groen AJ, Lilley KS and Trotter MWB. The Effect of Organelle Discovery upon Sub-Cellular Protein Localisation. *J Proteomics*. In Press.

**Examples**

```
## Not run:
library(pRolocdata)
data(tan2009r1)
pdres <- phenoDisco(tan2009r1, fcol = "PLSDA")
getPredictions(pdres, fcol = "pd", scol = NULL)
plot2D(pdres, fcol = "pd")

## End(Not run)
```

plot2D

*Plot organelle assignment data and results.***Description**

Generate 2 dimensional or feature distribution plots to illustrate localisation clusters. In plot2D, rows containing NA values are removed prior to dimension reduction.

**Usage**

```
plot2D(object, fcol = "markers", fpch,
       unknown = "unknown", dims = 1:2, alpha, score = 1,
       method = c("PCA", "MDS"), axsSwitch = FALSE,
       mirrorX = FALSE, mirrorY = FALSE, col, pch, cex,
       identify = FALSE, plot = TRUE, ...)
```

**Arguments**

object	An instance of class MSnSet.
fcol	Feature meta-data label (fData column name) defining the groups to be differentiated using different colours. Default is markers. Use NULL to suppress any colouring.
fpch	Feature meta-data label (fData column name) defining the groups to be differentiated using different point symbols.
unknown	A character (default is "unknown") defining how proteins of unknown localisation are labelled.
dims	A numeric of length 2 defining the dimensions to be plotted, i.e the PC/MDS axes.
alpha	A numeric defining the alpha channel (transparency) of the points, where $0 \leq \alpha \leq 1$ , 0 and 1 being completely transparent and opaque.
score	A numeric specifying the minimum organelle assignment score to consider features to be assigned an organelle. (not yet implemented).
method	One of PCA (default) or MDS, defining if dimensionality reduction is done using principal component analysis (see <a href="#">prcomp</a> ) or classical multidimensional scaling (see <a href="#">cmdscale</a> ).

axsSwitch	A logical indicating whether the axes should be switched.
mirrorX	A logical indicating whether the x axis should be mirrored?
mirrorY	A logical indicating whether the y axis should be mirrored?
col	A character of appropriate length defining colours.
pch	A character of appropriate length defining point character.
cex	Character expansion.
identify	A logical (default is TRUE) defining if user interaction will be expected to identify individual data points on the plot. See also <a href="#">identify</a> .
plot	A logical defining if the figure should be plotted. Useful when retrieving data only. Default is TRUE.
...	Additional parameters passed to plot and points.

**Value**

Used for its side effects of generating a plot. Invisibly returns the 2d data.

**Author(s)**

Laurent Gatto <lg390@cam.ac.uk>

**See Also**

[addLegend](#) to add a legend to plot2D figures and [plotDist](#) for alternative graphical representation of quantitative organelle proteomics data.

**Examples**

```
library("pRolocdata")
data(dunkley2006)
plot2D(dunkley2006, fcol = NULL)
plot2D(dunkley2006, fcol = "markers")
addLegend(dunkley2006,
          fcol = "markers",
          where = "topright",
          cex = 0.5, bty = "n", ncol = 3)
title(main = "plot2D example")
```

---

plotDist

*Plots the distribution of features across fractions*

---

**Description**

Produces a line plot showing the feature abundances across the fractions.

**Usage**

```
plotDist(object, markers, mcol = "steelblue",  
         pcol = "grey90", alpha = 0.3, fractions, ...)
```

**Arguments**

object	An instance of class MSnSet.
markers	A character, numeric or logical of appropriate length and or content used to subset object and define the organelle markers.
mcol	A character define the colour of the marker features. Default is "steelblue".
pcol	A character define the colour of the non-markers features. Default is "grey90".
alpha	A numeric defining the alpha channel (transparency) of the points, where $0 \leq \alpha \leq 1$ , 0 and 1 being completely transparent and opaque.
fractions	An optional character defining the phenoData variable to be used to label the fraction along the x axis. If missing, the phenoData variables are searched for a match to fraction. If no match is found, the fractions are labelled as numericals.
...	Additional parameters passed to <a href="#">plot</a> .

**Value**

Used for its side effect of producing a feature distribution plot. Invisibly returns NULL.

**Author(s)**

Laurent Gatto

**Examples**

```
library("pRolocdata")  
data(tan2009r1)  
j <- which(fData(tan2009r1)$markers == "mitochondrion")  
i <- which(fData(tan2009r1)$PLSDA == "mitochondrion")  
plotDist(tan2009r1[i, ],  
         markers = featureNames(tan2009r1)[j],  
         main = "Mitochondrion")
```

---

plsdaClassification    *plsda classification*

---

**Description**

Classification using the partial least square discriminant analysis algorithm.

**Usage**

```
plsdaClassification(object, assessRes,  
  scores = c("prediction", "all", "none"), ncomp,  
  fcol = "markers", ...)
```

**Arguments**

object	An instance of class "MSnSet".
assessRes	An instance of class "GenRegRes", as generated by <a href="#">plsdaOptimisation</a> .
scores	One of "prediction", "all" or "none" to report the score for the predicted class only, for all cluster or none.
ncomp	If assessRes is missing, a ncomp must be provided.
fcol	The feature meta-data containing marker definitions. Default is markers.
...	Additional parameters passed to <a href="#">plsda</a> from package <a href="#">caret</a> .

**Value**

An instance of class "MSnSet" with plsda and plsda.scores feature variables storing the classification results and scores respectively.

**Author(s)**

Laurent Gatto

**Examples**

```
## Not run:  
## not running this one for time considerations  
library(pRocdata)  
data(dunkley2006)  
## reducing parameter search space and iterations  
params <- plsdaOptimisation(dunkley2006, ncomp = c(3, 10), times = 2)  
params  
plot(params)  
f1Count(params)  
levelPlot(params)  
getParams(params)  
res <- plsdaClassification(dunkley2006, params)  
getPredictions(res, fcol = "plsda")  
getPredictions(res, fcol = "plsda", t = 0.75)  
plot2D(res, fcol = "plsda")  
  
## End(Not run)
```

---

plsdaOptimisation      *plsda parameter optimisation*

---

### Description

Classification parameter optimisation for the partial least square discriminant analysis algorithm.

### Usage

```
plsdaOptimisation(object, fcol = "markers", ncomp = 1:6,  
  times = 100, test.size = 0.2, xval = 5, fun = mean,  
  seed, verbose = TRUE, ...)
```

### Arguments

object	An instance of class " <a href="#">MSnSet</a> ".
fcol	The feature meta-data containing marker definitions. Default is markers.
ncomp	The hyper-parameter. Default values are 1:6.
times	The number of times internal cross-validation is performed. Default is 100.
test.size	The size of test data. Default is 0.2 (20 percent).
xval	The n-cross validation. Default is 5.
fun	The function used to summarise the xval macro F1 matrices.
seed	The optional random number generator seed.
verbose	A logical defining whether a progress bar is displayed.
...	Additional parameters passed to <a href="#">plsda</a> from package <a href="#">caret</a> .

### Value

An instance of class "[GenRegRes](#)".

### Author(s)

Laurent Gatto

### See Also

[plsdaClassification](#) and example therein.

---

rfClassification      *rf classification*

---

### Description

Classification using the random forest algorithm.

### Usage

```
rfClassification(object, assessRes,
  scores = c("prediction", "all", "none"), mtry,
  fcol = "markers", ...)
```

### Arguments

object	An instance of class "MSnSet".
assessRes	An instance of class "GenRegRes", as generated by <a href="#">rfOptimisation</a> .
scores	One of "prediction", "all" or "none" to report the score for the predicted class only, for all cluster or none.
mtry	If assessRes is missing, a mtry must be provided.
fcol	The feature meta-data containing marker definitions. Default is markers.
...	Additional parameters passed to <a href="#">randomForest</a> from package randomForest.

### Value

An instance of class "MSnSet" with rf and rf.scores feature variables storing the classification results and scores respectively.

### Author(s)

Laurent Gatto

### Examples

```
library(pRolocdata)
data(dunkley2006)
## reducing parameter search space and iterations
params <- rfOptimisation(dunkley2006, mtry = c(2, 5, 10), times = 3)
params
plot(params)
f1Count(params)
levelPlot(params)
getParams(params)
res <- rfClassification(dunkley2006, params)
getPredictions(res, fcol = "rf")
getPredictions(res, fcol = "rf", t = 0.75)
plot2D(res, fcol = "rf")
```



---

rfOptimisation      *svm parameter optimisation*

---

### Description

Classification parameter optimisation for the random forest algorithm.

### Usage

```
rfOptimisation(object, fcol = "markers", mtry = NULL,  
               times = 100, test.size = 0.2, xval = 5, fun = mean,  
               seed, verbose = TRUE, ...)
```

### Arguments

object	An instance of class " <a href="#">MSnSet</a> ".
fcol	The feature meta-data containing marker definitions. Default is markers.
mtry	The hyper-parameter. Default value is NULL.
times	The number of times internal cross-validation is performed. Default is 100.
test.size	The size of test data. Default is 0.2 (20 percent).
xval	The n-cross validation. Default is 5.
fun	The function used to summarise the xval macro F1 matrices.
seed	The optional random number generator seed.
verbose	A logical defining whether a progress bar is displayed.
...	Additional parameters passed to <a href="#">randomForest</a> from package randomForest.

### Value

An instance of class "[GenRegRes](#)".

### Author(s)

Laurent Gatto

### See Also

[rfClassification](#) and example therein.

---

svmClassification      *ksvm classification*

---

### Description

Classification using the support vector machine algorithm.

### Usage

```
svmClassification(object, assessRes,
  scores = c("prediction", "all", "none"), cost, sigma,
  fcol = "markers", ...)
```

### Arguments

object	An instance of class "MSnSet".
assessRes	An instance of class "GenRegRes", as generated by <a href="#">svmOptimisation</a> .
scores	One of "prediction", "all" or "none" to report the score for the predicted class only, for all cluster or none.
cost	If assessRes is missing, a cost must be provided.
sigma	If assessRes is missing, a sigma must be provided.
fcol	The feature meta-data containing marker definitions. Default is markers.
...	Additional parameters passed to <a href="#">svm</a> from package e1071.

### Value

An instance of class "MSnSet" with svm and svm.scores feature variables storing the classification results and scores respectively.

### Author(s)

Laurent Gatto

### Examples

```
library(pRolocdata)
data(dunkley2006)
## reducing parameter search space and iterations
params <- svmOptimisation(dunkley2006, cost = 2^seq(-2,2,2), sigma = 10^seq(-1, 1, 1), times = 3)
params
plot(params)
f1Count(params)
levelPlot(params)
getParams(params)
res <- svmClassification(dunkley2006, params)
getPredictions(res, fcol = "svm")
getPredictions(res, fcol = "svm", t = 0.75)
plot2D(res, fcol = "svm")
```

---

svmOptimisation      *svm parameter optimisation*

---

### Description

Classification parameter optimisation for the support vector machine algorithm.

### Usage

```
svmOptimisation(object, fcol = "markers",  
  cost = 2^(-4:4), sigma = 10^(-2:3), times = 100,  
  test.size = 0.2, xval = 5, fun = mean, seed,  
  verbose = TRUE, ...)
```

### Arguments

object	An instance of class " <a href="#">MSnSet</a> ".
fcol	The feature meta-data containing marker definitions. Default is markers.
cost	The hyper-parameter. Default values are $2^{-4:4}$ .
sigma	The hyper-parameter. Default values are $10^{-2:3}$ .
times	The number of times internal cross-validation is performed. Default is 100.
test.size	The size of test data. Default is 0.2 (20 percent).
xval	The n-cross validation. Default is 5.
fun	The function used to summarise the xval macro F1 matrices.
seed	The optional random number generator seed.
verbose	A logical defining whether a progress bar is displayed.
...	Additional parameters passed to <a href="#">svm</a> from package e1071.

### Value

An instance of class "[GenRegRes](#)".

### Author(s)

Laurent Gatto

### See Also

[svmClassification](#) and example therein.

# Index

## \*Topic **classes**

GenRegRes-class, 6

## \*Topic **methods**

chi2-methods, 4

exprsToRatios-methods, 6

MLearn-methods, 18

addLegend, 2, 28

addMarkers, 3

chi2, 5

chi2 (chi2-methods), 4

chi2, matrix, matrix-method  
(chi2-methods), 4

chi2, matrix, numeric-method  
(chi2-methods), 4

chi2, numeric, matrix-method  
(chi2-methods), 4

chi2, numeric, numeric-method  
(chi2-methods), 4

chi2-methods, 4

chol2inv, 23, 24

class:GenRegRes (GenRegRes-class), 6

cmdscale, 27

empPvalues, 4, 5

exprsToRatios (exprsToRatios-methods), 6

exprsToRatios, MSnSet-method  
(exprsToRatios-methods), 6

exprsToRatios-methods, 6

f1Count (GenRegRes-class), 6

f1Count, GenRegRes-method  
(GenRegRes-class), 6

filterNA, 26

GenRegRes, 11–14, 19–23, 25, 30–35

GenRegRes (GenRegRes-class), 6

GenRegRes-class, 6

getF1Scores (GenRegRes-class), 6

getF1Scores, GenRegRes-method  
(GenRegRes-class), 6

getMarkers, 8

getParams (GenRegRes-class), 6

getParams, GenRegRes-method  
(GenRegRes-class), 6

getPredictions, 8

getRegularisedParams (GenRegRes-class),  
6

getRegularisedParams, GenRegRes-method  
(GenRegRes-class), 6

getRegularizedParams (GenRegRes-class),  
6

getRegularizedParams, GenRegRes-method  
(GenRegRes-class), 6

getSeed (GenRegRes-class), 6

getSeed, GenRegRes-method  
(GenRegRes-class), 6

getStockcol, 9

getStockpch (getStockcol), 9

getUnknowncol (getStockcol), 9

getUnknownpch (getStockcol), 9

getWarnings (GenRegRes-class), 6

getWarnings, GenRegRes-method  
(GenRegRes-class), 6

ginv, 23, 24

identify, 28

knn, 11, 12

knnClassification, 11, 12

knnOptimisation, 11, 12

knnOptimization (knnOptimisation), 12

knnPrediction (knnClassification), 11

knnRegularisation, 6

knnRegularisation (knnOptimisation), 12

ksvm, 13, 14

ksvmClassification, 13, 14

ksvmOptimisation, 13, 14

ksvmOptimization (ksvmOptimisation), 14

- ksvmPrediction (ksvmClassification), 13
- ksvmRegularisation (ksvmOptimisation), 14
- legend, 3
- levelPlot (GenRegRes-class), 6
- levelPlot, GenRegRes-method (GenRegRes-class), 6
- makeNaData, 15
- makeNaData2 (makeNaData), 15
- minClassScore, 16
- minMarkers, 17
- MLearn, 18
- MLearn, formula, MSnSet, clusteringSchema, missing-method (MLearn-methods), 18
- MLearn, formula, MSnSet, learnerSchema, numeric-method (MLearn-methods), 18
- MLearn, formula, MSnSet, learnerSchema, xvalSpec-method (MLearn-methods), 18
- MLearn-methods, 18
- MLearnMSnSet (MLearn-methods), 18
- MSnSet, 6, 8, 9, 11–15, 17–24, 30–35
- MSnSetMLEan (MLearn-methods), 18
- naiveBayes, 19, 20
- nbClassification, 19, 20
- nbOptimisation, 19, 20
- nbOptimization (nbOptimisation), 20
- nbPrediction (nbClassification), 19
- nbRegularisation (nbOptimisation), 20
- nnet, 21, 22
- nnetClassification, 21, 22
- nnetOptimisation, 21, 22
- nnetOptimization (nnetOptimisation), 22
- nnetPrediction (nnetClassification), 21
- nnetRegularisation (nnetOptimisation), 22
- perTurboClassification, 23, 25
- perTurboOptimisation, 23, 24
- perTurboOptimization (perTurboOptimisation), 24
- phenoDisco, 25
- plot, 29
- plot, GenRegRes, missing-method (GenRegRes-class), 6
- plot2D, 2, 27
- plotDist, 28, 28
- plsda, 30, 31
- plsdaClassification, 29, 31
- plsdaOptimisation, 30, 31
- plsdaOptimization (plsdaOptimisation), 31
- plsdaPrediction (plsdaClassification), 29
- plsdaRegularisation, 6
- plsdaRegularisation (plsdaOptimisation), 31
- prcomp, 27
- randomForest, 32, 33
- rfClassification, 32, 33
- rfOptimisation, 32, 33
- rfOptimization (rfOptimisation), 33
- rfPrediction (rfClassification), 32
- rfRegularisation (rfOptimisation), 33
- setStockcol (getStockcol), 9
- setStockpch (getStockcol), 9
- setUnknowncol (getStockcol), 9
- setUnknownpch (getStockcol), 9
- show, GenRegRes-method (GenRegRes-class), 6
- solve, 23, 24
- svd, 23, 24
- svm, 34, 35
- svmClassification, 34, 35
- svmOptimisation, 34, 35
- svmOptimization (svmOptimisation), 35
- svmPrediction (svmClassification), 34
- svmRegularisation, 6, 23
- svmRegularisation (svmOptimisation), 35
- whichNA (makeNaData), 15
- xvalSpec, 18