Package 'pRoloc'

April 10, 2015

Type Package

Title A unifying bioinformatics framework for spatial proteomics

Version 1.6.2

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 quantitiative mass spectrometry data to infer protein sub-cellular localisation.

Depends R (>= 2.15), MSnbase (>= 1.13.3), MLInterfaces (>= 1.37.1), methods, Rcpp (>= 0.10.3), BiocParallel

Imports mclust (>= 4.3), caret, e1071, sampling, class, kernlab, lattice, nnet, randomForest, proxy, FNN, BiocGenerics, stats, RColorBrewer, scales, MASS, knitr, mvtnorm

Suggests testthat, pRolocdata, roxygen2, synapter, xtable

LinkingTo Rcpp, RcppArmadillo

License GPL-2

VignetteBuilder knitr

Video https://www.youtube.com/playlist?list=PLvIXxpatSLA2loV5Srs2VBpJIYUlVJ4ow

BugReports https://github.com/lgatto/pRoloc/issues

biocViews Proteomics, MassSpectrometry, Classification, Clustering, QualityControl

R topics documented:

`addLegend` *Adds a legend*

Description

Adds a legend to a `plot2D` figure.

Usage

```
addLegend(object, fcol = "markers", where = c("other", "bottomright",
  "bottom", "bottomleft", "left", "topleft", "top", "topright", "right",
  "center"), 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 legend.
col	A character defining point colours.
...	Additional parameters passed to legend.

Details

The function has been updated in version 1.3.6 to recycle the default colours when more organelle classes are provided. See plot2D for details.

Value

Invisibly returns NULL

Author(s)

Laurent Gatto

---

addLegend_v1 *Adds a legend*

Description

Adds a legend to a plot2D_v1 figure.

Usage

```
addLegend_v1(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 legend.
col	A character defining point colours.
...	Additional parameters passed to legend.

Value

Invisibly returns NULL

Author(s)

Laurent Gatto

---

addMarkers *Adds markers to the data*

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. Alternatively, a markers named vector as provided by the pRolocmarkers function can also be used.

Usage

```
addMarkers(object, markers, mcol = "markers", fcol, verbose = TRUE)
```

Arguments

object	An instance of class MSnSet.
markers	A character with the name the markers' csv file or a named character of markers as provided by pRolocmarkers.
mcol	A character of length 1 defining the feature variable label for the newly added markers. Default is "markers".
fcol	An optional feature variable to be used to match against the markers. If missing, the feature names are used.
verbose	A logical indicating if number of markers and marker table should be printed to the console.

Details

It is essential to assure that `featureNames(object)` (or `fcol`, see below) and marker names (first column) match, i.e. the same feature identifiers and case fold are used.

Value

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

Author(s)

Laurent Gatto

Examples

```
library("pRolocdata")
data(dunkley2006)
atha <- pRolocmarkers("atha")
try(addMarkers(dunkley2006, atha)) ## markers already exists
fData(dunkley2006)$markers.org <- fData(dunkley2006)$markers
fData(dunkley2006)$markers <- NULL
marked <- addMarkers(dunkley2006, atha)
fvarLabels(marked)
## if makers already exists
marked <- addMarkers(marked, atha, mcol = "markers2")
fvarLabels(marked)
stopifnot(all.equal(fData(marked)$markers, fData(marked)$markers2))
plot2D(marked)
addLegend(marked, where = "topleft", cex = .7)
```

---

chi2-methods *The PCP 'chi square' method*

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/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 $Chi\hat{}2$ 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 nrow(corMatrix) protein relative intensities to be compares against the marker.
n	The number of iterations.
...	Additional parameters to be passed to chi2.

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 dimenstions `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 outter 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` *Returns the organelle markers in an 'MSnSet'*

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", names = TRUE, verbose = TRUE)
```

Arguments

object	An instance of class "MSnSet".
fcol	The name of the markers column in the featureData slot. Default is markers.
names	A logical indicating if the markers vector should be named.
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

See Also

testMarkers and minMarkers

Examples

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

---

getPredictions *Returns the predictions in an 'MSnSet'*

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.
t	The score threshold. Predictions with score < t are set to 'unknown'. Default is 0. It is also possible to define thresholds for each prediction class, in which case, t is a named numeric with names exactly matching the unique prediction class names.

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

Examples

```
library("pRolocdata")
data(dunkley2006)
res <- svmClassification(dunkley2006, fcol = "pd.markers",
                         sigma = 0.1, cost = 0.5)
fData(res)$svm[500:510]
fData(res)$svm.scores[500:510]
getPredictions(res, fcol = "svm", t = 0) ## all predictions
getPredictions(res, fcol = "svm", t = .9) ## single threshold
## 50% top predictions per class
(ts <- tapply(fData(res)$svm.scores, fData(res)$svm, median))
getPredictions(res, fcol = "svm", t = ts)
## 50% top predictions per class, ignoring marker scores
ts <- tapply(fData(res)$svm.scores, fData(res)$svm,
             function(x) {
                 scr <- median(x[x != 1])
                 ifelse(is.na(scr), 1, scr)
             })
ts
getPredictions(res, fcol = "svm", t = ts)
```

---

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()
```

`highlightOnPlot` *Highlight features of interest on a plot2D figure*

Description

Highlights a set of features of interest given as a `FeaturesOfInterest` instance on a PCA plot produced by codeplot2D. If none of the features of interest are found in the `MSnset`'s `featureNames`, an error is thrown.

Usage

```
highlightOnPlot(object, foi, args = list(), ...)
```

Arguments

`object`	The main dataset described as an `MSnSet` or a matrix with the coordinates of the features on the PCA plot produced (and invisibly returned) by `plot2D`.
`foi`	An instance of `FeaturesOfInterest`.
`args`	A named list of arguments to be passed to `plot2D` if the PCA coordinates are to be calculated. Ignored if the PCA coordinates are passed directly, i.e. `object` is a `matrix`.
`...`	Additional parameters passed to `points`.

Value

NULL; used for its side effects.

Author(s)

Laurent Gatto

Examples

```
library("pRolocdata")
data("tan2009r1")
x <- FeaturesOfInterest(description = "A test set of features of interest",
                        fnames = featureNames(tan2009r1)[1:10],
                        object = tan2009r1)
.pca <- plot2D(tan2009r1)
highlightOnPlot(.pca, x, col = "red")
```

```
highlightOnPlot(tan2009r1, x, col = "red", cex = 1.5)

.pca <- plot2D(tan2009r1, dims = c(1, 3))
highlightOnPlot(.pca, x, pch = "+")
highlightOnPlot(tan2009r1, x, args = list(dims = c(1, 3)))

plot2D(tan2009r1, mirrorX = TRUE)
highlightOnPlot(.pca, x, pch = "+", args = list(mirrorX = TRUE))
```

---

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 knnOptimisation.
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 knn from package class.

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 = seq(3, 15, 2), times = 100,
  test.size = 0.2, xval = 5, fun = mean, seed, verbose = TRUE, ...)
```

Arguments

object	An instance of class "MSnSet".
fcol	The feature meta-data containing marker definitions. Default is markers.
k	The hyper-parameter. Default values are seq(3, 15, 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 knn from package class.

Details

Note that when performance scores precision, recall and (macro) F1 are calculated, any NA values are replaced by 0. This decision is motivated by the fact that any class that would have either a NA precision or recall would result in an NA F1 score and, eventually, a NA macro F1 (i.e. mean(F1)). Replacing NAs by 0s leads to F1 values of 0 and a reduced yet defined final macro F1 score.

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 ksvmOptimisation.
`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 ksvm 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(pRolocdata)
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)
getParams(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 "`MSnSet`".
`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 `ksvm` from package `kernlab`.

Details

Note that when performance scores precision, recall and (macro) F1 are calculated, any NA values are replaced by 0. This decision is motivated by the fact that any class that would have either a NA precision or recall would result in an NA F1 score and, eventually, a NA macro F1 (i.e. mean(F1)). Replacing NAs by 0s leads to F1 values of 0 and a reduced yet defined final macro F1 score.

Value

An instance of class "GenRegRes".

Author(s)

Laurent Gatto

See Also

ksvmClassification and example therein.

`lopims` *A complete LOPIMS pipeline*

Description

The function processes MSe data using the synergise function of the synapter package and combines resulting Synapter instances into one "MSnSet" and organelle marker data is added as a feature-level annotation variable.

Usage

```
lopims(hdmsedir = "HDMSE", msedir = "MSE", pep3ddir = "pep3D", fastafile,
  markerfile, mfdr = 0.025, ...)
```

Arguments

hdmsedir	A character identifying the directory containing the HDMSe final peptide files. Default is HDMSe.
msedir	A character identifying the directory containing the MSe final peptide files. Default is MSe.
pep3ddir	A character identifying the directory containing the MSe pep 3D files. Default is pep3D.
fastafile	A character identifying the protein fasta database. Default is to use the fasta file in the current directory. If several such files exist, the function reports an error.
markerfile	A character identifying the marker file (see details for format). Default is to use a csv file starting with marker in the current directory. If several such files exist, the function reports an error.
mfdr	The master FDR value. Default is 0.025.
...	Additional paramters passed to synergise.

Details

The LOPIMS pipeline is composed of 5 steps:

1. The HDMSe final peptide files are used to compute false discovery rates uppon all possible combinations of HDMSe final peptides files and the best combination smaller or equal to mfdr is chosen. See estimateMasterFdr for details. The corresponding master run is then created as descibed in makeMaster. (function lopims1)
2. Each MSe/pep3D pair is processed using the HDMSe master file using synergise. (function lopims2)
3. The respective peptide-level synergise output objects are converted and combined into an single "MSnSet" instance. (function lopims3)
4. Protein-level quantitation is inferred as follows. For each protein, a reference sample/fraction is chosen based on the number of missing values (NA). If several samples have a same minimal number of NAs, ties are broken using the sum of counts. The peptides that do not display any missing values for each (frac_i, frac_ref) pair are summed and the ratio is reported (see pRoloc:::refNormMeanOfNonNAPepSum for details). (function lopims4)
5. The markers defined in the markerfile are collated as feature meta-data in the markers variable. See addMarkers for details. (function lopims5)

Intermediate synergise reports as well as resulting objects are stored in a LOPIMS_pipeline directory. For details, please refer to the synapter vignette and reference papers.

Value

An instance of class "MSnSet" with protein level quantitation and respective organelle markers.

Author(s)

Laurent Gatto

References

Improving qualitative and quantitative performance for MSE-based label free proteomics. N.J. Bond, P.V. Shliaha, K.S. Lilley and L. Gatto, Journal of Proteome Research, 2013 (in press).

The Effects of Travelling Wave Ion Mobility Separation on Data Independent Acquisition in Proteomics Studies, P.V. Shliaha, N.J. Bond, L. Gatto and K.S. Lilley, Journal of Proteome Research, 2013 (in press).

MSnbase-an R/Bioconductor package for isobaric tagged mass spectrometry data visualization, processing and quantitation. L. Gatto and KS. Lilley. Bioinformatics. 2012 Jan 15;28(2):288-9. doi: 10.1093/bioinformatics/btr645. Epub 2011 Nov 22. PubMed PMID: 22113085.

makeNaData *Create a data with missing values*

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 assignmed.
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))
```

`markerMSnSet` *Extract marker/unknown subsets*

Description

These function extract the marker or unknown proteins into a new `MSnSet`.

Usage

```
markerMSnSet(object, fcol = "markers")

unknownMSnSet(object, fcol = "markers")
```

Arguments

object	An instance of class MSnSet
fcol	The name of the feature data column, that will be used to separate the markers from the proteins of unknown localisation (with fData(object)[, fcol] == "unknown"). Default is to use "markers".

Value

An new MSnSet with marker/unknown proteins only.

Author(s)

Laurent Gatto

See Also

sampleMSnSet testMSnSet

Examples

```
library("pRolocdata")
data(dunkley2006)
mrk <- markerMSnSet(dunkley2006)
unk <- unknownMSnSet(dunkley2006)
dim(dunkley2006)
dim(mrk)
dim(unk)
table(fData(dunkley2006)$markers)
table(fData(mrk)$markers)
table(fData(unk)$markers)
```

---

minClassScore *Updates classes based on prediction scores*

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(pRolocdata)
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(pRolocdata)
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 *nb classification*

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 "MSnSet".
assessRes	An instance of class "GenRegRes", as generated by nbOptimisation.
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 naiveBayes from package e1071.

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

object	An instance of class "MSnSet".
fcol	The feature meta-data containing marker definitions. Default is markers.
laplace	The hyper-parameter. Default values are seq(0, 5, 0.5).
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 naiveBayes from package e1071.

Details

Note that when performance scores precision, recall and (macro) F1 are calculated, any NA values are replaced by 0. This decision is motivated by the fact that any class that would have either a NA precision or recall would result in an NA F1 score and, eventually, a NA macro F1 (i.e. mean(F1)). Replacing NAs by 0s leads to F1 values of 0 and a reduced yet defined final macro F1 score.

Value

An instance of class "GenRegRes".

Author(s)

Laurent Gatto

See Also

nbClassification and example therein.

nndist-methods *Nearest neighbour distances*

Description

Methods computing the nearest neighbour indices and distances for `matrix` and `MSnSet` instances.

Methods

`signature(object = "matrix", k = "numeric", dist = "character", ...)` Calculates indices and distances to the `k` (default is 3) nearest neighbours of each feature (row) in the input matrix `object`. The distance `dist` can be either of `"euclidean"` or `"mahalanobis"`. Additional parameters can be passed to the internal function `FNN::get.knn`. Output is a matrix with `2 * k` columns and `nrow(object)` rows.

`signature(object = "MSnSet", k = "numeric", dist = "character", ...)` As above, but for an `MSnSet` input. The indices and distances to the `k` nearest neighbours are added to the object's feature metadata.

`signature(object = "matrix", query = "matrix", k = "numeric", ...)` If two `matrix` instances are provided as input, the `k` (default is 3) indices and distances of the nearest neighbours of `query` in `object` are returned as a matrix of dimensions `2 * k` by `nrow(query)`. Additional parameters are passed to `FNN::get.knnx`. Only euclidean distance is available.

Examples

```
library("pRolocdata")
data(dunkley2006)

## Using a matrix as input
m <- exprs(dunkley2006)
m[1:4, 1:3]
head(nndist(m, k = 5))
tail(nndist(m[1:100, ], k = 2, dist = "mahalanobis"))

## Same as above for MSnSet
d <- nndist(dunkley2006, k = 5)
head(fData(d))

d <- nndist(dunkley2006[1:100, ], k = 2, dist = "mahalanobis")
tail(fData(d))

## Using a query
nndist(m[1:100, ], m[101:110, ], k = 2)
```

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 nnetOptimisation.
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 nnet 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 "MSnSet".
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 nnet from package nnet.

Details

Note that when performance scores precision, recall and (macro) F1 are calculated, any NA values are replaced by 0. This decision is motivated by the fact that any class that would have either a NA precision or recall would result in an NA F1 score and, eventually, a NA macro F1 (i.e. mean(F1)). Replacing NAs by 0s leads to F1 values of 0 and a reduced yet defined final macro F1 score.

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 "MSnSet".
assessRes	An instance of class "GenRegRes", as generated by svmRegularisation.
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 perTurboOptimisation for details.
sigma	If assessRes is missing, a sigma must be provided. See perTurboOptimisation for details.
inv	The type of algorithm used to invert the matrix. Values are : "Inversion Cholesky" (chol2inv), "Moore Penrose" (ginv), "solve" (solve), "svd" (svd). 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 Wieczorek

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 "MSnSet".
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" (chol2inv), "Moore Penrose" (ginv), "solve" (solve), "svd" (svd). 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.

Details

Note that when performance scores precision, recall and (macro) F1 are calculated, any NA values are replaced by 0. This decision is motivated by the fact that any class that would have either a NA precision or recall would result in an NA F1 score and, eventually, a NA macro F1 (i.e. mean(F1)). Replacing NAs by 0s leads to F1 values of 0 and a reduced yet defined final macro F1 score.

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, ndims = 2,
  modelNames = mclust.options("emModelNames"), G = 1:9, BPPARAM, tmpfile,
  seed, verbose = TRUE)
```

Arguments

object	An instance of class MSnSet.
fcol	A character indicating the organellar markers column name in feature metadata. 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.
ndims	Number of principal components to use as input for the disocvery analysis. Default is 2. Added in version 1.3.9.
modelNames	A vector of characters indicating the models to be fitted in the EM phase of clustering using Mclust. The help file for mclustModelNames describes the available models. Default model names are c("EII", "VII", "EEI","VEI", "EVI", "VVI", "EEE", "EEV" as returned by mclust.options("emModelNames"). Note that using all these possible models substantially increases the running time. Legacy models are c("EEE","EEV","VEV","VVV"), i.e. only ellipsoidal models.
G	An integer vector specifying the numbers of mixture components (clusters) for which the BIC is to be calculated. The default is G=1:9 (as in Mclust).
BPPARAM	Support for parallel processing using the BiocParallel infrastructure. When missing (default), the default registered BiocParallelParam parameters are used. Alternatively, one can pass a valid BiocParallelParam parameter instance: SnowParam, MulticoreParam, DoparParam, ... see the BiocParallel package for details. To revert to the origianl serial implementation, use NULL.
tmpfile	An optional character to save a temporary MSnSet after each iteration. Ignored if missing. This is useful for long runs to track phenotypes and possibly kill the run when convergence is observed. If the run completes, the temporary file is deleted before returning the final result.
seed	An optional numeric of length 1 specifing the random number generator seed to be used. Only relevant when executed in serialised mode with BPPARAM = NULL. See BPPARAM for details.
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 features with missing values using the filterNA method.

A parallel implementation, relying on the BiocParallel package, has been added in version 1.3.9. See the BPPARAM arguent for details.

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.

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. 2013 Aug 2;88:129-40. doi: 10.1016/j.jprot.2013.02.019. Epub 2013 Mar 21. PubMed PMID: 23523639.

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 localistation clusters. In plot2D, rows containing NA values are removed prior to dimention reduction.

Usage

```
plot2D(object, fcol = "markers", fpch, unknown = "unknown", dims = 1:2,
  score = 1, method = c("PCA", "MDS", "kpca", "scree"), methargs,
  axsSwitch = FALSE, mirrorX = FALSE, mirrorY = FALSE, col, pch, cex,
  index = FALSE, idx.cex = 0.75, 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	Featre meta-data label (fData column name) desining 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. Always 1:2 for MDS.
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), MDS or kpca, defining if dimensionality reduction is done using principal component analysis (see prcomp), classical multidimensional scaling (see cmdscale) or kernel PCA (see kernlab::kpca).
methargs	A list of arguments to be passed when method is called. If missing, the data will be scaled and centred prior to PCA.
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.
index	A logical (default is FALSE), indicating of the feature indices should be plotted on top of the symbols.
idx.cex	A numeric specifying the character expansion (default is 0.75) for the feature indices. Only relevant when index is TRUE.
identify	A logical (default is TRUE) defining if user interaction will be expected to identify individual data points on the plot. See also identify.
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.

Details

Note that plot2D has been update in version 1.3.6 to support more more orgnalle classes than colours defined in getStockcol. In such cases, the defauly colours are recycled using the default plotting characters defined in getStockpch. See the example for an illustration. The alpha argument is also depreciated in version 1.3.6. Use setStockcol to set colours with transparency instead. See example below.

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 = NULL, method = "kpca")
plot2D(dunkley2006, fcol = NULL, method = "kpca",
       methargs = list(kpar = list(sigma = 1)))
plot2D(dunkley2006, fcol = "markers")
addLegend(dunkley2006,
          fcol = "markers",
          where = "topright",
          cex = 0.5, bty = "n", ncol = 3)
title(main = "plot2D example")
## Using transparent colours
setStockcol(paste0(getStockcol(), "80"))
plot2D(dunkley2006, fcol = "markers")
## New behavious in 1.3.6 when not enough colours
setStockcol(c("blue", "red", "green"))
getStockcol() ## only 3 colours to be recycled
getMarkers(dunkley2006)
plot2D(dunkley2006)
```

---

plot2D_v1 *Plot organelle assignment data and results.*

Description

This is the documentation for the pre-v 1.3.6 function.

Usage

```
plot2D_v1(object, fcol = "markers", fpch, unknown = "unknown", dims = 1:2,
  alpha, score = 1, outliers = TRUE, method = c("PCA", "MDS", "kpca"),
  methargs, axsSwitch = FALSE, mirrorX = FALSE, mirrorY = FALSE, col, pch,
  cex, index = FALSE, idx.cex = 0.75, 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	Featre meta-data label (fData column name) desining 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. Always 1:2 for MDS.
alpha	A numeric defining the alpha channel (transparency) of the points, where 0 <= alpha <= 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).
outliers	A logical specifying whether outliers should be plotted or ignored (default is TRUE, i.e. all points are plotted). Useful when the presence of outliers masks the structure of the rest of the data. Outliers are defined by the 2.5 and 97.5 percentiles.
method	One of PCA (default), MDS or kpca, defining if dimensionality reduction is done using principal component analysis (see prcomp), classical multidimensional scaling (see cmdscale) or kernel PCA (see kernlab::kpca).
methargs	A list of arguments to be passed when method is called. If missing, the data will be scaled and centred prior to PCA.
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.
index	A logical (default is FALSE), indicating if the feature indices should be plotted on top of the symbols.
idx.cex	A numeric specifying the character expansion (default is 0.75) for the feature indices. Only relevant when index is TRUE.
identify	A logical (default is TRUE) defining if user interaction will be expected to identify individual data points on the plot. See also identify.
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.

Details

Generate 2 dimensional or feature distribution plots to illustrate localistation clusters. In plot2D_v1, rows containing NA values are removed prior to dimention reduction.

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_v1 figures and plotDist for alternative graphical representation of quantitative organelle proteomics data.

Examples

```
library("pRolocdata")
data(dunkley2006)
pRoloc:::plot2D_v1(dunkley2006, fcol = NULL)
pRoloc:::plot2D_v1(dunkley2006, fcol = NULL, method = "kpca")
pRoloc:::plot2D_v1(dunkley2006, fcol = NULL, method = "kpca",
                   methargs = list(kpar = list(sigma = 1)))
pRoloc:::plot2D_v1(dunkley2006, fcol = "markers")
pRoloc:::addLegend_v1(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, lty = 1, fractions, ylim, ...)
```

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 <= alpha <= 1, 0 and 1 being completely transparent and opaque.
lty	Vector of line types for the marker profiles. Default is 1 (solid). See par for details.
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.
ylim	A numeric vector of length 2, giving the y coordinates range.
...	Additional parameters passed to plot.

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])
title("Mitochondrion")
```

---

plsdaClassification *plsda classification*

Description

Classification using the partial least square distcriminant 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 plsdaOptimisation.
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 plsda from package caret.

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(pRolocdata)
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 = 0.9)
plot2D(res, fcol = "plsda")

## End(Not run)
```

---

plsdaOptimisation *plsda parameter optimisation*

Description

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

Usage

```
plsdaOptimisation(object, fcol = "markers", ncomp = 2:6, times = 100,
  test.size = 0.2, xval = 5, fun = mean, seed, verbose = TRUE, ...)
```

Arguments

object	An instance of class "MSnSet".
fcol	The feature meta-data containing marker definitions. Default is markers.
ncomp	The hyper-parameter. Default values are 2:6.
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 plsda from package caret.

Details

Note that when performance scores precision, recall and (macro) F1 are calculated, any NA values are replaced by 0. This decision is motivated by the fact that any class that would have either a NA precision or recall would result in an NA F1 score and, eventually, a NA macro F1 (i.e. mean(F1)). Replacing NAs by 0s leads to F1 values of 0 and a reduced yet defined final macro F1 score.

Value

An instance of class "GenRegRes".

Author(s)

Laurent Gatto

See Also

plsdaClassification and example therein.

pRolocmarkers *Organelle markers*

Description

This function retrieves a list of organelle markers or, if no species is provided, prints a description of available marker sets. The markers can be added to and MSnSet using the addMarkers function.

Usage

```
pRolocmarkers(species)
```

Arguments

species	The species of interest.

Details

The markers have been contributed by various members of the Cambridge Centre for Proteomics, in particular Dan Nightingale for yeast, Dr Andy Christoforou for human, Dr Arnoud Groen for Arabodopsis and Dr Claire Mulvey for mouse. In addition, original (curated) markers from the pRolocdata datasets have been extracted (see pRolocdata for details and references). Curation involved verification of publicly available subcellular localisation annotation based on the curators knowledge of the organelles/proteins considered and tracing the original statement in the literature.

These markers are provided as a starting point to generate reliable sets of organelle markers but still need to be verified against any new data in the light of the quantitative data and the study conditions.

Value

Prints a description of the available marker lists if species is missing or a named character with organelle markers.

Author(s)

Laurent Gatto

Examples

```
pRolocmarkers()
table(pRolocmarkers("atha"))
table(pRolocmarkers("hsap"))
```

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 rfOptimisation.
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 randomForest 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 "MSnSet".
`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 `randomForest` from package `randomForest`.

Details

Note that when performance scores precision, recall and (macro) F1 are calculated, any NA values are replaced by 0. This decision is motivated by the fact that any class that would have either a NA precision or recall would result in an NA F1 score and, eventually, a NA macro F1 (i.e. mean(F1)). Replacing NAs by 0s leads to F1 values of 0 and a reduced yet defined final macro F1 score.

Value

An instance of class "GenRegRes".

Author(s)

Laurent Gatto

See Also

`rfClassification` and example therein.

sampleMSnSet *Extract a stratified sample of an* MSnSet

Description

This function extracts a stratified sample of an MSnSet.

Usage

```
sampleMSnSet(object, fcol = "markers", size = 0.2, seed)
```

Arguments

object	An instance of class "MSnSet"
fcol	The feature meta-data column name containing the marker definitions on which the MSnSet will be stratified. Default is markers.
size	The size of the stratified sample to be extracted. Default is 0.2 (20 percent).
seed	The optional random number generator seed.

Value

A stratified sample (according to the defined fcol) which is an instance of class "MSnSet".

Author(s)

Lisa Breckels

See Also

testMSnSet unknownMSnSet markerMSnSet

Examples

```
library(pRolocdata)
data(tan2009r1)
dim(tan2009r1)
mySample <- sampleMSnSet(tan2009r1, fcol = "PLSDA")
dim(mySample)
```

svmClassification *svm 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 svmOptimisation.
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 svm 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^(-3:2), times = 100, test.size = 0.2, xval = 5,
  fun = mean, seed, verbose = TRUE, ...)
```

Arguments

object	An instance of class "MSnSet".
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 svm from package e1071.

Details

Note that when performance scores precision, recall and (macro) F1 are calculated, any NA values are replaced by 0. This decision is motivated by the fact that any class that would have either a NA precision or recall would result in an NA F1 score and, eventually, a NA macro F1 (i.e. mean(F1)). Replacing NAs by 0s leads to F1 values of 0 and a reduced yet defined final macro F1 score.

Value

An instance of class "GenRegRes".

Author(s)

Laurent Gatto

See Also

svmClassification and example therein.

`testMarkers` *Tests marker class sizes*

Description

Tests if the marker class sizes are large enough for the parameter optimisation scheme, i.e. the size is greater that `xval + n`, where the default `xval` is 5 and `n` is 2. If the test is unsuccessful, a warning is thrown.

Usage

```
testMarkers(object, xval = 5, n = 2, fcol = "markers", error = FALSE)
```

Arguments

`object`	An instance of class `"MSnSet"`.
`xval`	The number cross-validation partitions. See the `xval` argument in the parameter optimisation function(s). Default is 5.
`n`	Number of additional examples.
`fcol`	The name of the prediction column in the `featureData` slot. Default is `"markers"`.
`error`	A `logical` specifying if an error should be thown, instead of a warning.

Details

In case the test indicates that a class contains too few examples, it is advised to either add some or, if not possible, to remove the class altogether (see `minMarkers`) as the parameter optimisation is likely to fail or, at least, produce unreliable results for that class.

Value

If successfull, the test invisibly returns `NULL`. Else, it invisibly returns the names of the classes that have too few examples.

Author(s)

Laurent Gatto

See Also

`getMarkers` and `minMarkers`

Examples

```
library("pRolocdata")
data(dunkley2006)
getMarkers(dunkley2006)
testMarkers(dunkley2006)
toosmall <- testMarkers(dunkley2006, xval = 15)
toosmall
try(testMarkers(dunkley2006, xval = 15, error = TRUE))
```

`testMSnSet` *Create a stratified 'test'* `MSnSet`

Description

This function creates a stratified 'test' `MSnSet` which can be used for algorihtmic development. A `"MSnSet"` containing only the marker proteins, as defined in `fcol`, is returned with a new feature data column appended called `test` in which a stratified subset of these markers has been relabelled as 'unknowns'.

Usage

```
testMSnSet(object, fcol = "markers", size = 0.2, seed)
```

Arguments

`object`	An instance of class `"MSnSet"`
`fcol`	The feature meta-data column name containing the marker definitions on which the data will be stratified. Default is `markers`.
`size`	The size of the data set to be extracted. Default is 0.2 (20 percent).
`seed`	The optional random number generator seed.

Value

An instance of class `"MSnSet"` which contains only the proteins that have a labelled localisation i.e. the marker proteins, as defined in `fcol` and a new column in the feature data slot called `test` which has part of the labels relabelled as "unknown" class (the number of proteins renamed as "unknown" is according to the parameter size).

Author(s)

Lisa Breckels

See Also

`sampleMSnSet` `unknownMSnSet` `markerMSnSet`

Examples

```
library(pRolocdata)
data(tan2009r1)
sample <- testMSnSet(tan2009r1)
getMarkers(sample, "test")
all(dim(sample) == dim(markerMSnSet(tan2009r1)))
```

---

undocumented *Undocumented/unexported entries*

Description

This is just a dummy entry for methods from unexported classes that generate warnings during package checking.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Index