Package ‘pRoloc’

November 21, 2016

Type Package

Title A unifying bioinformatics framework for spatial proteomics

Version 1.14.4

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 (>= 1.19.20), MLInterfaces (>= 1.37.1), methods, Rcpp (>= 0.10.3), BiocParallel

Imports Biobase, mclust (>= 4.3), caret, e1071, sampling, class, kernlab, lattice, nnet, randomForest, proxy, FNN, BiocGenerics, stats, dendextend, RColorBrewer, scales, MASS, knitr, mvtnorm, gtools, plyr, ggplot2, biomaRt, utils, grDevices, graphics

Suggests testthat, rmarkdown, pRolocdata (>= 1.9.4), roxygen2, synapter, xtable, tsne, hexbin, rgl, BiocStyle, hpar (>= 1.15.3), dplyr, GO.db, AnnotationDbi

LinkingTo Rcpp, RcppArmadillo

License GPL-2

VignetteBuilder knitr

Video https://www.youtube.com/playlist?list=PLvIXxpatSLA2loV5Sr2VBpJlYUIVJ4ow

URL https://github.com/lgatto/pRoloc

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

biocViews Proteomics, MassSpectrometry, Classification, Clustering, QualityControl

Collate AllGenerics.R machinelearning-framework.R
machinelearning-framework-theta.R machinelearning-utils.R
machinelearning-functions-nnet.R machinelearning-functions-PerTurbo.R

RoxygenNote  5.0.1

NeedsCompilation  yes

R topics documented:

pRoloc-package  .......................................................... 3
addGoAnnotations ......................................................... 4
addLegend  ..................................................................... 6
addMarkers ................................................................. 6
AnnotationParams-class .................................................. 8
checkFeatureNamesOverlap ............................................. 9
checkFvarOverlap .......................................................... 10
chi2-methods ............................................................... 11
classWeights .............................................................. 12
clustDist ................................................................. 13
ClustDist-class ............................................................ 14
ClustDistList-class ...................................................... 15
empPvalues ................................................................. 17
exprsToRatios-methods ................................................ 18
fDataToUnknown ........................................................ 18
filterBinMSnSet .......................................................... 19
filterMaxMarkers ......................................................... 20
filterMinMarkers ........................................................ 21
filterZeroCols ............................................................ 21
GenRegRes-class .......................................................... 22
getGOFromFeatures ..................................................... 23
getMarkerClasses ......................................................... 24
getMarkers ............................................................... 25
getNormDist .............................................................. 26
getPredictions ........................................................... 27
goldToTerm ............................................................... 28
highlightOnPlot ........................................................ 29
knnClassification ......................................................... 30
knnOptimisation ........................................................ 31
knntlClassification ...................................................... 32
knntlOptimisation ....................................................... 33
ksvmClassification ...................................................... 35
ksvmOptimisation ....................................................... 36
lopims ................................................................. 37
makeGoSet ............................................................. 38
markerMSnSet .......................................................... 39
MartInstance-class ...................................................... 40
minMarkers ............................................................. 40
### Description

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

More details about the package provided in the following vignettes:

**pRoloc-ml**  An overview of the machine learning techniques available in the pRoloc package.

**pRoloc-tutorial**  The main pRoloc tutorial, providing a hands-on introduction to the package, including data requirements, visualisation, clustering, classification and the application of semi-supervised machine learning.

**pRoloc-transfer-learning**  Description of a transfer learning algorithm for spatial proteomics.


If you have questions, want to report a bug or share suggestions, please file an issue at [https://github.com/lgatto/MSnbase/issues](https://github.com/lgatto/MSnbase/issues), contact me directly or ask a question on the Bioconductor support forum [https://support.bioconductor.org/](https://support.bioconductor.org/).

Author(s)

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

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

References


See Also

The underlying infrastructure to store and manipulate the quantitative data is implemented in the MSnbase package. See [MSnbase](https://m.snbase.org) to get started.

### addGoAnnotations

**Add GO annotations**

**Description**

Adds GO annotations to the feature data
addGoAnnotations

Usage

addGoAnnotations(object, params, evidence, useID = FALSE,
  fcol = "GOAnnotations", ...)

Arguments

object  An instance of class MSnSet.
params  An instance of class AnnotationParams. If missing, getAnnotationParams
  will be used.
evidence GO evidence filtering.
useID   Logical. Should GO term names or identifiers be used? If TRUE, identifiers
  will be used. If FALSE GO term names will be used.
fcol    Character. Name of the matrix of annotations to be added to the fData default
  is GOAnnotations
...
Other arguments passed to makeGoSet

Value

An updated MSnSet with new feature data column called GOAnnotations containing a matrix of
GO annotations

Author(s)

Lisa M Breckels

Examples

library(pRolocdata)
data(dunkley2006)
par <- setAnnotationParams(inputs =
  c("Arabidopsis thaliana genes",
    "Gene stable ID"))
## add protein sets/annotation information
xx <- addGoAnnotations(dunkley2006, par)
dim(fData(xx)$GOAnnotations)

## filter sets
xx <- filterMinMarkers(xx, n = 50)
dim(fData(xx)$GOAnnotations)
xx <- filterMaxMarkers(xx, p = .25)
dim(fData(xx)$GOAnnotations)

## Subset for specific protein sets
sub <- subsetMarkers(xx, keep = c("vacuole"))

## Order protein sets
res <- orderGoAnnotations(xx, k = 1:3, p = 1/3, verbose = FALSE)
if (interactive()) {
pRolocVis(res, fcol = "GOAnnotations")
}
Description

Adds a legend to a plot2D figure.

Usage

addLegend(object, fcol = "markers", where = c("bottomleft", "bottom", "bottomright", "left", "topleft", "top", "topright", "right", "center", "other"), col, bty = "n", ...)

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 "bottomleft" (default), "bottomright", "topleft", "topright" or "other" 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.
bty Box type, as in legend. Default is set to "n".
... 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

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.
addMarkers

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

See Also

See pRolocmarkers for a list of spatial markers and markers for details about markers encoding.

Examples

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

Description

Class to store annotation parameters to automatically query a Biomart server, retrieve relevant annotation for a set of features of interest using, for example `getGOFromFeatures` and `makeGoSet`.

Objects from the Class

Objects can be created and set with the `setAnnotationParams` function. Object are created by calling without any arguments `setAnnotationParams()`, which will open an interactive interface. Depending on the value of "many.graphics" option, a graphical of a text-based menu will open (the text interface can be forced by setting the graphics argument to FALSE: `setAnnotationParams(graphics = FALSE)`). The menu will allow to select the species of interest first and the type of features (ENSEMBL gene identifier, Entrez id, ...) second.

The species that are available are those for which ENSEMBL data is available in Biomart and have a set of attributes of interest available. The compatible identifiers for downstream queries are then automatically filtered and displayed for user selection.

It is also possible to pass a parameter `inputs`, a character vector of length 2 containing a pattern uniquely matching the species of interest (in position 1) and a patterns uniquely matching the feature types (in position 2). If the matches are not unique, an error will be thrown.

A new instance of the AnnotationParams will be created to enable easy and automatic query of the Mart instance. The instance is invisibly returned and stored in a global variable in the pRoloc package's private environment for automatic retrieval. If a variable containing an AnnotationParams instance is already available, it can be set globally by passing it as argument to the `setAnnotationParams` function. Globally set AnnotationParams instances can be accessed with the `getAnnotationParams` function.

See the `pRoloc-theta` vignette for details.

Slots

- `mart`: Object of class "Mart" from the `biomaRt` package.
- `martname`: Object of class "character" with the name of the mart instance.
- `dataset`: Object of class "character" with the data set of the mart instance.
- `filter`: Object of class "character" with the filter to be used when querying the mart instance.
- `date`: Object of class "character" indicating when the current instance was created.
- `biomaRtVersion`: Object of class "character" with the `biomaRt` version used to create the AnnotationParams instance.
- `__classVersion__`: Object of class "Versions" with the version of the AnnotationParams class of the current instance.

Methods

- `show` signature(object = "AnnotationParams"): to display objects.
checkFeatureNamesOverlap

Author(s)
Laurent Gatto <lg390@cam.ac.uk>

See Also
getGOFromFeatures, makeGoSet and the pRoloc-theta vignette.

Examples

data(andy2011params)
andy2011params
data(dunkley2006params)
dunkley2006params

try(setAnnotationParams(inputs = c("nomatch1", "nomatch2")))
setAnnotationParams(inputs = c("Homo sapiens", "UniProt/Swissprot Accession"))
getAnnotationParams()

Value
Invisibly returns a named list of common markers, unique x markers, unique y markers in, common unknowns, unique x unknowns and unique y unknowns.

Author(s)
Laurent Gatto
Examples

```r
library("pRolocdata")
data(andy2011)
data(andy2011goCC)
checkFeatureNamesOverlap(andy2011, andy2011goCC)
featureNames(andy2011goCC)[1] <- "ABC"
res <- checkFeatureNamesOverlap(andy2011, andy2011goCC)
res$markersX
res$markersY
```

---

**checkFvarOverlap**

**Compare a feature variable overlap**

Description

Extracts qualitative feature variables from two MSnSet instances and compares with a contingency table.

Usage

```r
checkFvarOverlap(x, y, fcolx = "markers", fcoly, verbose = TRUE)
```

Arguments

- `x`: An MSnSet instance.
- `y`: An MSnSet instance.
- `fcolx`: The feature variable to separate unknown (fData(y)$coly == "unknown") from the marker features in the x object.
- `fcoly`: As `fcolx`, for the y object. If missing, the value of `fcolx` is used.
- `verbose`: If TRUE (default), the contingency table of the the feature variables is printed out.

Value

Invisibly returns a named list with the values of the diagonal, upper and lower triangles of the contingency table.

Author(s)

Laurent Gatto

Examples

```r
library("pRolocdata")
data(dunkley2006)
res <- checkFvarOverlap(dunkley2006, dunkley2006, "markers", "markers.orig")
str(res)
```
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, \( \sum \frac{(x_i - x_p)^2}{x_p} \), 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 \texttt{chi2} methods implement this idea and compute such \( \chi^2 \) values for sets of proteins.

Methods

\texttt{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 \( \sum((y-x)^2/x) \). \texttt{na.rm} defines if missing values (NA and NaN) should be removed prior to summation. \texttt{fun} defines how to summarise the \( \chi^2 \) values; default, \texttt{NULL}, does not combine the \( \chi^2 \) values.

\texttt{signature(x = "matrix", y = "numeric", method = "character", na.rm = "logical")}

Computes \( nrow(x) \) \( \chi^2 \) values, for all the \( (x, y) \) pairs. See above for the other arguments.

\texttt{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.

\texttt{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


See Also

\texttt{empPvalues}
classWeights

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)

classWeights

Calculate class weights

Description

Calculates class weights to be used for parameter optimisation and classification such as svmOptimisation or svmClassification - see the pRoloc tutorial vignette for an example. The weights are calculated for all non-unknown classes the inverse of the number of observations.

Usage

classWeights(object, fcol = "markers")

Arguments

object An instance of class MSnSet
fcol The name of the features to be weighted

Value

A table of class weights

Author(s)

Laurent Gatto

Examples

library("pRolocdata")
data(hyperLOPIT2015)
classWeights(hyperLOPIT2015)
data(dunkley2006)
classWeights(dunkley2006)
clustDist

Pairwise Distance Computation for Protein Information Sets

Description

This function computes the mean (normalised) pairwise distances for pre-defined sets of proteins.

Usage

clustDist(object, k = 1:5, fcol = "GOAnnotations", n = 5, 
verbose = TRUE, seed)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An instance of class &quot;MSnSet&quot;.</td>
</tr>
<tr>
<td>k</td>
<td>The number of clusters to try fitting to the protein set. Default is k = 1:5.</td>
</tr>
<tr>
<td>fcol</td>
<td>The feature meta-data containing matrix of protein sets/ marker definitions. Default is GOAnnotations.</td>
</tr>
<tr>
<td>n</td>
<td>The minimum number of proteins per set. If protein sets contain less than n instances they will be ignored. Default is 5.</td>
</tr>
<tr>
<td>verbose</td>
<td>A logical defining whether a progress bar is displayed.</td>
</tr>
<tr>
<td>seed</td>
<td>An optional seed for the random number generator.</td>
</tr>
</tbody>
</table>

Details

The input to the function is a MSnSet dataset containing a matrix appended to the feature data slot identifying the membership of protein instances to a pre-defined set(s) e.g. a specific Gene Ontology term etc.

For each protein set, the clustDist function (i) extracts all instances belonging to the set, (ii) using the kmeans algorithm fits and tests k = c(1:5) (default) cluster components to each set, (iii) calculates the mean pairwise distance for each k tested.

Note: currently distances are calculated in Euclidean space, but other distance metrics will be supported in the future).

The output is a list of ClustDist objects, one per information cluster. The ClustDist class summarises the algorithm information such as the number of k’s tested for the kmeans, and mean and normalised pairwise Euclidean distances per number of component clusters tested. See ?ClustDist for more details.

Value

An instance of "ClustDistList" containing a "ClustDist" instance for every protein set, which summarises the algorithm information such as the number of k’s tested for the kmeans, and mean and normalised pairwise Euclidean distances per number of component clusters tested.

Author(s)

Lisa Breckels
See Also

For class definitions see "ClustDistList" and "ClustDist".

Examples

```r
library(pRolocdata)
data(dunkley2006)
par <- setAnnotationParams(inputs =
  c("Arabidopsis thaliana genes",
"Gene stable ID"))
## add protein sets/annotation information
xx <- addGoAnnotations(dunkley2006, par)
## filter
xx <- filterMinMarkers(xx, n = 50)
x <- filterMaxMarkers(xx, p = .25)
## get distances for protein sets
dd <- clustDist(xx)
## plot clusters for first 'ClustDist' object
## in the 'ClustDistList'
plot(dd[[1]], xx)
## plot distances for all protein sets
plot(dd)
## Extract normalised distances
## Normalise by n^1/3
minDist <- getNormDist(dd, p = 1/3)
## Get new order according to lowest distance
o <- order(minDist)
## Re-order GOAnnotations
fData(xx)$GOAnnotations <- fData(xx)$GOAnnotations[, o]
if (interactive()) {
pRolocVis(xx, fcol = "GOAnnotations")
}
```

Description

The ClustDist summaries algorithm information, from running the clustDist function, such as the number of k’s tested for the kmeans, and mean and normalised pairwise (Euclidean) distances per number of component clusters tested.

Objects from the Class

Object of this class are created with the clustDist function.

Slots

- **k**: Object of class "numeric" storing the number of k clusters tested.
- **dist**: Object of class "list" storing the list of distance matrices.
- **term**: Object of class "character" describing GO term name.
- **id**: Object of class "character" describing the GO term ID.
**ClustDistList-class**

**nrow**: Object of class "numeric" showing the number of instances in the set

**clustsz**: Object of class "list" describing the number of instances for each cluster for each k tested

**components**: Object of class "vector" storing the class membership of each protein for each k tested.

**fcol**: Object of class "character" showing the feature column name in the corresponding MSnSet where the protein set information is stored.

**Methods**

- **plot**  Plots the kmeans clustering results.
- **show**  Shows the object.

**Author(s)**

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

**Examples**

```r
showClass("ClustDist")

library(\'pRolocdata\'

data(dunkley2006)
par <- setAnnotationParams(inputs =
  c("Arabidopsis thaliana genes",
    "Gene stable ID"))

## add protein set/annotation information
xx <- addGoAnnotations(dunkley2006, par)

## filter
xx <- filterMinMarkers(xx, n = 50)
xx <- filterMaxMarkers(xx, p = .25)

## get distances for protein sets
dd <- clustDist(xx)

## plot clusters for first 'ClustDist' object
## in the 'ClustDistList'
plot(dd[[1]], xx)

## plot distances for all protein sets
plot(dd)
```

---

**ClustDistList-class**  
Storing multiple `ClustDist` instances

**Description**

A class for storing lists of `ClustDist` instances.
Objects from the Class

Object of this class are created with the clustDist function.

Slots

x: Object of class list containing valid ClustDist instances.
log: Object of class list containing an object creation log, containing among other elements the call that generated the object.
__classVersion__: The version of the instance. For development purposes only.

Methods

"[" Extracts a single ClustDist at position.
"[" Extracts one of more ClustDists as ClustDistList.
length Returns the number of ClustDists.
names Returns the names of ClustDists, if available. The replacement method is also available.
show Display the object by printing a short summary.
lapply(x, FUN, ...) Apply function FUN to each element of the input x. If the application of
FUN returns and clustDist, then the return value is an ClustDistList, otherwise a list.
plot Plots a boxplot of the distance results per protein set.

Author(s)

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

Examples

library('pRolocdata')
data(dunkley2006)
par <- setAnnotationParams(inputs =
    c("Arabidopsis thaliana genes",
      "Gene stable ID"))

## add protein set/annotation information
xx <- addGoAnnotations(dunkley2006, par)

## filter
xx <- filterMinMarkers(xx, n = 50)
xx <- filterMaxMarkers(xx, p = .25)

## get distances for protein sets
dd <- clustDist(xx)

## plot distances for all protein sets
plot(dd)

names(dd)

## Extract first 4 ClustDist objects of the ClustDistList
dd[1:4]

## Extract 1st ClustDist object
dd[[1]]
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 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


See Also

- chi2 for Chi\(^2\) calculation.

Examples

```r
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**

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

**fDataToUnknown  Update a feature variable**

**Description**

This function replaces a string or regular expression in a feature variable using the `sub` function.

**Usage**

```r
fDataToUnknown(object, fcol = "markers", from = "^\$", to = "unknown", ...)
```

**Arguments**

- **object**
  - An instance of class MSnSet.

- **fcol**
  - Feature variable to be modified. Default is "markers". If NULL, all feature variables will updated.

- **from**
  - A character defining the string or regular expression of the pattern to be replaced. Default is the empty string, i.e. the regular expression "^\$". See `sub` for details. If NA, then NA values are replaced by to.

- **to**
  - A replacement for matched pattern. Default is "unknown". See `sub` for details.

- **...**
  - Additional arguments passed to `sub`.

**Value**

An updated MSnSet.
filterBinMSnSet

Author(s)

Laurent Gatto

Examples

library("pRolocdata")
data(dunkley2006)
getMarkers(dunkley2006, "markers")
dunkley2006 <- fDataToUnknown(dunkley2006,
from = "unknown", to = "unassigned")
getMarkers(dunkley2006, "markers")

filterBinMSnSet object, MARGIN = 2, t, q, verbose = TRUE)

Arguments

object An MSnSet
MARGIN 1 or 2. Default is 2.
t Rows/columns that have t or less 1s, it will be filtered out. When t and q are missing, default is to use t = 1.
q If a row has a higher quantile than defined by q, it will be filtered out.
verbose A logical defining of a message is to be printed. Default is TRUE.

Value

A filtered MSnSet.

Author(s)

Laurent Gatto

See Also

zerosInBinMSnSet, filterZeroCols, filterZeroRows.
Examples

```r
set.seed(1)
m <- matrix(sample(0:1, 25, replace=TRUE), 5)
m[1, ] <- 0
m[, 1] <- 0
rownames(m) <- colnames(m) <- letters[1:5]
fd <- data.frame(row.names = letters[1:5])
x <- MSnSet(exprs = m, fData = fd, pData = fd)
exprs(x)
## Remove columns with no 1s
exprs(filterBinMSnSet(x, MARGIN = 2, t = 0))
## Remove columns with one 1 or less
exprs(filterBinMSnSet(x, MARGIN = 2, t = 1))
## Remove columns with two 1s or less
exprs(filterBinMSnSet(x, MARGIN = 2, t = 2))
## Remove columns with three 1s
exprs(filterBinMSnSet(x, MARGIN = 2, t = 3))
## Remove columns that have half or less of 1s
exprs(filterBinMSnSet(x, MARGIN = 2, q = 0.5))
```

**filterMaxMarkers**

Removes class/annotation information from a matrix of candidate markers that appear in the fData.

Description

Removes annotation information that contain more than a certain number/percentage of proteins

Usage

`filterMaxMarkers(object, n, p = 0.2, fcol = "GOAnnotations", verbose = TRUE)`

Arguments

- **object**: An instance of class MSnSet.
- **n**: Maximum number of proteins allowed per class/information term.
- **p**: Maximum percentage of proteins per column. Default is 0.2 i.e. remove columns that have information for greater than 20 of the total number of proteins in the dataset (note: this is useful for example, if information is GO terms, for removing very general and uninformative terms).
- **fcol**: The name of the matrix of marker information. Default is GOAnnotations.
- **verbose**: The name of the matrix of marker information. Default is GOAnnotations.

Value

An updated MSnSet

See Also

`addGoAnnotations` and example therein.
### filterMinMarkers

Removes class/annotation information from a matrix of candidate markers that appear in the `fData`.

**Description**

Removes annotation information that contain less that a certain number/percentage of proteins

**Usage**

```r
filterMinMarkers(object, n = 10, p, fcol = "GOAnnotations", verbose = TRUE)
```

**Arguments**

- `object`: An instance of class `MSnSet`.
- `n`: Minimum number of proteins allowed per column. Default is 10.
- `p`: Minimum percentage of proteins per column.
- `fcol`: The name of the matrix of marker information. Default is `GOAnnotations`.
- `verbose`: Number of marker candidates retained after filtering.

**Value**

An updated `MSnSet`.

**Author(s)**

Lisa M Breckels

**See Also**

- `addGoAnnotations` and example therein.

### filterZeroCols

Remove 0 columns/rows

**Description**

Removes all assay data columns/rows that are composed of only 0, i.e. have a `colSum/rowSum` of 0.

**Usage**

```r
filterZeroCols(object, verbose = TRUE)

filterZeroRows(object, verbose = TRUE)
```

**Arguments**

- `object`: A `MSnSet` object.
- `verbose`: Print a message with the number of filtered out columns/row (if any).
Value
An MSnSet.

Author(s)
Laurent Gatto

Examples
library("pRolocdata")
data(andy2011goCC)
any(colSums(exprs(andy2011goCC)) == 0)
exprs(andy2011goCC)[, 1:5] <- 0
ncol(andy2011goCC)
ncol(filterZeroCols(andy2011goCC))

\[
\text{GenRegRes-class}
\]

Class "GenRegRes" and "ThetaRegRes"

Description
Regularisation framework containers.

Objects from the Class
Object of this class are created with the respective regularisation function: \texttt{knnOptimisation}, \texttt{svmOptimisation}, \texttt{plsdaOptimisation}, \texttt{knntlOptimisation}, ...

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 hyper-parameters + 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.
Only in ThetaRegRes:
predictions: A list of predictions for the optimisation iterations.
otherWeights: Alternative best theta weights: a vector per iterations, NULL if no other best weights were found.
Methods

getF1Scores Returns a matrix of F1 scores for the optimisation parameters.

f1Count signature(object = "GenRegRes", t = "numeric") and signature(object = "ThetaRegRes", 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.

getParams Returns the best parameters. It is however strongly recommended to inspect the optimisation results. For a ThetaRegRes optimisation result, the method to chose the best parameters can be "median" (default) or "mean" (the median or mean of the best weights is chosen), "max" (the first weights with the highest macro-F1 score, considering that multiple max scoring combinations are possible) or "count" (the observed weight that get the maximum number of observations, see f1Count). The favourP argument can be used to prioritise weights that favour the primary data (i.e. heigh weights). See favourPrimary below.

getSeed Returns the seed used for the optimisation run.

getWarnings signature(object = "GenRegRes"): Returns a vector of recorded warnings.

levelPlot signature(object = "GenRegRes"): Plots a heatmap of of the optimisation results.

plot Plots the optimisation results.

show Shows the object.

Other functions

Only for ThetaRegRes:

combineThetaRegRes(object) Takes a list of ThetaRegRes instances to be combined and returns a new ThetaRegRes instance.

favourPrimary(primary, auxiliary, object, verbose = TRUE) Takes the primary and auxiliary data sources (two MSnSet instances) and a ThetaRegRes object and returns and updated ThetaRegRes instance containing best parameters/weigths (see the getParams function) favouring the primary data when multiple best theta weights are available.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

showClass("GenRegRes")
showClass("ThetaRegRes")

getGOFromFeatures Retrieve GO terms for feature names

Description

The function pulls the gene ontology (GO) terms for a set of feature names.
Usage

getGOFromFeatures(id, namespace = "cellular_component", evidence = NULL, params = NULL, verbose = FALSE, nmax = 500)

Arguments

id          An character with feature names to be pulled from biomart. If and MSnSet is provided, then featureNames(id) is used.
namespace   The GO namespace. One of biological_process, cellular_component (default) or molecular_function.
evidence    The GO evidence code. See showGOEvidenceCodes for details. If NULL (default), no filtering based on the evidence code is performed.
params      An instance of class "AnnotationParams".
verbose     A logical defining verbosity of the function. Default is FALSE.
nmax        As described in https://support.bioconductor.org/p/86358/, the Biomart result can be unreliable for large queries. This argument splits the input in chunks of length nmax (default is 500). If set to NULL, the query is performed in full.

Value

A data.frame with relevant GO terms.

Author(s)

Laurent Gatto

Examples

library(pRolocdata)
data(dunkley2006)
data(dunkley2006params)
dunkley2006params
fn <- featureNames(dunkley2006)[1:5]
getGOFromFeatures(fn, params = dunkley2006params)

getMarkerClasses

Returns the organelle classes in an 'MSnSet'

Description

Convenience accessor to the organelle classes in an 'MSnSet'. This function returns the organelle classes of an MSnSet instance. As a side effect, it prints out the classes.

Usage

getMarkerClasses(object, fcol = "markers", ...)
**getMarkers**

### Arguments

- **object**: An instance of class "MSnSet".
- **fcol**: The name of the markers column in the featureData slot. Default is markers.
- **...**: Additional parameters passed to `sort` from the base package.

### Value

A character vector of the organelle classes in the data.

### Author(s)

Lisa Breckels and Laurent Gatto

### See Also

`getMarkers` to extract the marker proteins. See `markers` for details about spatial markers storage and encoding.

### Examples

```r
library("pRolocdata")
data(dunkley2006)
organelles <- getMarkerClasses(dunkley2006)
## same if markers encoded as a matrix
dunkley2006 <- mrkVecToMat(dunkley2006, mfcol = "Markers")
orGanelles2 <- getMarkerClasses(dunkley2006, fcol = "Markers")
stopifnot(all.equal(organelles, organelles2))
```

---

**getMarkers**

*Get 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

```r
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. Ignored if markers are encoded as a matrix.
- **verbose**: If TRUE, a marker table is printed and the markers are returned invisibly. If FALSE, the markers are returned.
getNormDist

Description
This function computes and outputs normalised distances from a "ClustDistList" object.

Usage
getNormDist(object, p = 1/3)

Arguments
object An instance of class "ClustDistList".
p The normalisation factor. Default is 1/3.

Value
An numeric of normalised distances, one per protein set in the ClustDistList.

Author(s)
Lisa Breckels

See Also
"ClustDistList", "ClustDist" and examples in clustDist.
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, mcol = "markers", 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.
- **mcol**: The feature meta data column containing the labelled training data.
- **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

An instance of class "MSnSet" with fcol.pred feature variable storing the prediction results according to the chosen threshold.

Author(s)

Laurent Gatto and Lisa Breckels

See Also

- **orgQuants** for calculating organelle-specific thresholds.

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 <- orgQuants(res, fcol = "svm", t = .5)
getPredictions(res, fcol = "svm", t = ts)

---

**goIdToTerm**

Convert GO ids to/from terms

**Description**

Converts GO identifiers to/from GO terms, either explicitly or by checking if (any items in) the input contains "GO:"

**Usage**

```r
goIdToTerm(x, names = TRUE, keepNA = TRUE)
goTermToId(x, names = TRUE, keepNA = TRUE)
flipGoTermId(x, names = TRUE, keepNA = TRUE)
prettyGoTermId(x)
```

**Arguments**

- `x` A character of GO ids or terms.
- `names` Should a named character be returned? Default is `TRUE`.
- `keepNA` Should any GO term/id names that are missing or obsolete be replaced with a `NA`? Default is `TRUE`. If `FALSE` then the GO term/id names is kept.

**Value**

A character of GO terms (ids) if `x` were ids (terms).

**Author(s)**

Laurent Gatto

**Examples**

```r
goIdToTerm("GO:0000001")
goIdToTerm("GO:0000001", names = FALSE)
goIdToTerm(c("GO:0000001", "novalid"))
goIdToTerm(c("GO:0000001", "GO:0000002", "notvalid"))
goTermToId("mitochondrion inheritance")
goTermToId("mitochondrion inheritance", name = FALSE)
goTermToId(c("mitochondrion inheritance", "notvalid"))
prettyGoTermId("mitochondrion inheritance")
prettyGoTermId("GO:0000001")
flipGoTermId("mitochondrion inheritance")
flipGoTermId("GO:0000001")
flipGoTermId("GO:0000001", names = FALSE)
```
highlightOnPlot  

**Highlight features of interest on a spatial proteomics plot**

**Description**

Highlights a set of features of interest given as a `FeaturesOfInterest` instance on a PCA plot produced by `codeplot2D` or `plot3D`. If none of the features of interest are found in the `MSnset`'s `featureNames`, an warning is thrown.

**Usage**

```r
highlightOnPlot(object, foi, labels, args = list(), ...)  
highlightOnPlot3D(object, foi, labels, args = list(), radius = 0.1 * 3, ...)
```

**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`, or, alternatively, a character of feature names.

- `labels`  
  A character of length 1 with a feature variable name to be used to label the features of interest. This is only valid if `object` is an `MSnSet`. Alternatively, if `TRUE`, then `featureNames(object)` (or `rownames(object)` if `object` is a matrix) are used. Default is missing, which does not add any labels.

- `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` or `text` (when `labels` is `TRUE`) when adding to `plot2D`, or `spheres3d` or `text3d` when adding the `plot3D`.

- `radius`  
  Radius of the spheres to be added to the visualisation produced by `plot3D`. Default is 0.3 (i.e `plot3D`'s `radius1 * 3`), to emphasise the features with regard to unknown (`radius1 = 0.1`) and marker (`radius1 * 2`) features.

**Value**

NULL; used for its side effects.

**Author(s)**

Laurent Gatto

**Examples**

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

```r
## using FeaturesOfInterest or feature names
```
```r
par(mfrow = c(2, 1))
plot2D(tan2009r1)
highlightOnPlot(tan2009r1, x)
plot2D(tan2009r1)
highlightOnPlot(tan2009r1, featureNames(tan2009r1)[1:10])

.pca <- plot2D(tan2009r1)
head(.pca)
highlightOnPlot(.pca, x, col = "red")
highlightOnPlot(tan2009r1, x, col = "red", cex = 1.5)
highlightOnPlot(tan2009r1, x, labels = TRUE)

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

.pca2 <- plot2D(tan2009r1, mirrorX = TRUE, dims = c(1, 3))
## previous pca matrix, need to mirror X axis
highlightOnPlot(.pca, x, pch = "*", args = list(mirrorX = TRUE))
## new pca matrix, with X mirrors (and 1st and 3rd PCs)
highlightOnPlot(.pca2, x, col = "red")

plot2D(tan2009r1)
highlightOnPlot(tan2009r1, x)
highlightOnPlot(tan2009r1, x, labels = TRUE, pos = 3)
highlightOnPlot(tan2009r1, x, labels = "Flybase.Symbol", pos = 1)

## in 3 dimensions
plot3D(tan2009r1, radius1 = 0.05)
highlightOnPlot3D(tan2009r1, x, labels = TRUE)
highlightOnPlot3D(tan2009r1, x)
```

---

**knnClassification**  
**knn classification**

**Description**

Classification using for the k-nearest neighbours algorithm.

**Usage**

```r
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)
flCount(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.

knntlClassification

knntlClassification

knn transfer learning classification

Description

Classification using a variation of the KNN implementation of Wu and Dietterich’s transfer learning schema

Usage

knntlClassification(primary, auxiliary, fcol = "markers", bestTheta, k, scores = c("prediction", "all", "none"), seed)

Arguments

primary An instance of class "MSnSet".
auxiliary An instance of class "MSnSet".
fcol The feature meta-data containing marker definitions. Default is markers.
bestTheta Best theta vector as output from knntlOptimisation, see knntlOptimisation for details
k Numeric vector of length 2, containing the best k parameters to use for the primary and auxiliary datasets. If k k is not specified it will be calculated internally.
scores One of "prediction", "all" or "none" to report the score for the predicted class only, for all cluster or none.
seed The optional random number generator seed.

Value

A character vector of the classifications for the unknowns
knntlOptimisation

Author(s)
Lisa Breckels

See Also
knntlOptimisation

Examples

library(pRolocdata)
data(andy2011)
data(andy2011goCC)

## reducing calculation time of k by pre-running knnOptimisation
x <- c(andy2011, andy2011goCC)
k <- lapply(x, function(z)
    knnOptimisation(z, times=5,
        fcol = "markers.orig",
        verbose = FALSE))
k <- sapply(k, function(z) getParams(z))
k

## reducing parameter search with theta = 1,
## weights of only 1 or 0 will be considered
opt <- knntlOptimisation(andy2011, andy2011goCC,
    fcol = "markers.orig",
    times = 2,
    by = 1, k = k)

opt
th <- getParams(opt)
plot(opt)
res <- knntlClassification(andy2011, andy2011goCC,
    fcol = "markers.orig", th, k)
res

knntlOptimisation

theta parameter optimisation

Description
Classification parameter optimisation for the KNN implementation of Wu and Dietterich’s transfer learning schema

Usage
knntlOptimisation(primary, auxiliary, fcol = "markers", k, times = 50,
    test.size = 0.2, xval = 5, by = 0.5, length.out, th, xfolds,
    BPPARAM = BiocParallel::bpparam(), method = "Breckels", log = FALSE,
    seed)
**Arguments**

- **primary**: An instance of class "MSnSet".
- **auxiliary**: An instance of class "MSnSet".
- **fcol**: The feature meta-data containing marker definitions. Default is markers.
- **k**: Numeric vector of length 2, containing the best k parameters to use for the primary (k[1]) and auxiliary (k[2]) datasets. See knnOptimisation for generating best k.
- **times**: The number of times cross-validation is performed. Default is 50.
- **test.size**: The size of test (validation) data. Default is 0.2 (20 percent).
- **xval**: The number of rounds of cross-validation to perform.
- **by**: The increment for theta, must be one of c(1, 0.5, 0.25, 0.2, 0.15, 0.1, 0.05)
- **length.out**: Alternative to using by parameter. Specifies the desired length of the sequence of theta to test.
- **th**: A matrix of theta values to test for each class as generated from the function thetas, the number of columns should be equal to the number of classes contained in fcol. Note: columns will be ordered according to getMarkerClasses(primary, fcol). This argument is only valid if the default method 'Breckels' is used.
- **xfolds**: Option to pass specific folds for the cross validation.
- **BPPARAM**: Required for parallelisation. If not specified selects a default BiocParallelParam, from global options or, if that fails, the most recently registered() back-end.
- **method**: The k-NN transfer learning method to use. The default is 'Breckels' as described in the Breckels et al (2016). If 'Wu' is specified then the original method implemented Wu and Dietterich (2004) is implemented.
- **log**: A logical defining whether logging should be enabled. Default is FALSE. Note that logging produces considerably bigger objects.
- **seed**: The optional random number generator seed.

**Details**


**Value**

A list of containing the theta combinations tested, associated macro F1 score and accuracy for each combination over each round (specified by times).

**Author(s)**

Lisa Breckels

**References**


ksvmClassification

See Also

knntlClassification and example therein.

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

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.
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 upon 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 described 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**


---

**makeGoSet**

*Creates a GO feature MSnSet*

**Description**

Creates a new "MSnSet" instance populated with a GO term binary matrix based on an original object.

**Usage**

```r
makeGoSet(object, params, namespace = "cellular_component", evidence = NULL)
```

**Arguments**

- **object**: An instance of class "MSnSet" or a character of feature names.
- **params**: An instance of class "AnnotationParams", compatible with featureNames(object)'s format.
- **namespace**: The ontology name space. One or several of "biological_process", "cellular_component" or "molecular_function".
- **evidence**: GO evidence filtering.

**Value**

A new "MSnSet" with the GO terms for the respective features in the original object.
**markerMSnSet**

**Author(s)**

Laurent Gatto

**Examples**

```r
library("pRolocdata")
data(dunkley2006)
data(dunkley2006params)
goset <- makeGoSet(dunkley2006[1:10, ],
                  dunkley2006params)
goset
exprs(goset)[1:10, 1:5]
image(goset)
```

---

**markerMSnSet** | *Extract marker/unknown subsets*

**Description**

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

**Usage**

```r
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. When the markers are encoded as vectors, features of unknown localisation are defined as `fData(object)[, fcol] == "unknown"`. For matrix-encoded markers, unlabelled proteins are defined as `rowSums(fData(object)[, fcol]) == 0`. Default is "markers".

**Value**

An new MSnSet with marker/unknown proteins only.

**Author(s)**

Laurent Gatto

**See Also**

`sampleMSnSet testMSnSet` and `markers` for markers encoding.
**Examples**

```r
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)
## matrix-encoded markers
dunkley2006 <- mrkVecToMat(dunkley2006)
dim(markerMSnSet(dunkley2006, "Markers"))
stopifnot(all.equal(featureNames(markerMSnSet(dunkley2006, "Markers")),
 featureNames(markerMSnSet(dunkley2006, "markers"))))
dim(unknownMSnSet(dunkley2006, "Markers"))
stopifnot(all.equal(featureNames(unknownMSnSet(dunkley2006, "Markers")),
 featureNames(unknownMSnSet(dunkley2006, "markers"))))
```

---

**MartInstance-class**

*Class* "MartInstance"

**Description**

Internal infrastructure to query/handle several individual mart instance. See MartInterface.R for details.

**Author(s)**

Laurent Gatto <lg390@cam.ac.uk>

---

**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**

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

**Arguments**

- `object` An instance of class "MSnSet".
- `n` Minimum 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

See Also

gPredictions to filter based on classification scores.

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’ MLearn 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.
move2Ds

Displays a spatial proteomics animation

Description

Given two MSnSet instances of one MSnSetList with at least two items, this function produces an animation that shows the transition from the first data to the second.

Usage

move2Ds(object, pcol, fcol = "markers", n = 25, hl)

Arguments

object
An linkS4class{MSnSet} or a MSnSetList. In the latter case, only the two first elements of the list will be used for plotting and the others will be silently ignored.

pcol
If object is an MSnSet, a factor or the name of a phenotype variable (phenoData slot) defining how to split the single MSnSet into two or more data sets. Ignored if object is a MSnSetList.

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.

n
Number of frames, Default is 25.

hl
An optional instance of class linkS4class{FeaturesOfInterest} to track features of interest.

Value

Used for its side effect of producing a short animation.

Author(s)

Laurent Gatto

See Also

plot2Ds to a single figure with the two datasets.

Examples

library("pRolocdata")
data(dunkley2006)

## Create a relevant MSnSetList using the dunkley2006 data
xx <- split(dunkley2006, "replicate")
xx1 <- xx[[1]]
xx2 <- xx[[2]]
fData(xx1)$markers[374] <- "Golgi"
fData(xx2)$markers[412] <- "unknown"
xx@x[[1]] <- xx1
xx@x[[2]] <- xx2

## The features we want to track
foi <- FeaturesOfInterest(description = "test",
                           fnames = featureNames(xx[[1]])[c(374, 412)])

## (1) visualise each experiment separately
par(mfrow = c(2, 1))
plot2D(xx[[1]], main = "condition A")
highlightOnPlot(xx[[1]], foi)
plot2D(xx[[2]], mirrorY = TRUE, main = "condition B")
highlightOnPlot(xx[[2]], foi, args = list(mirrorY = TRUE))

## (2) plot both data on the same plot
par(mfrow = c(1, 1))
tmp <- plot2Ds(xx)
highlightOnPlot(data1(tmp), foi, lwd = 2)
highlightOnPlot(data2(tmp), foi, pch = 5, lwd = 2)

## (3) create an animation
move2Ds(xx, pcol = "replicate")
move2Ds(xx, pcol = "replicate", hl = foi)

---

mrkHClust  
**Draw a dendrogram of subcellular clusters**

### Description

This function calculates an average protein profile for each marker class (proteins of unknown localisation are ignored) and then generates a dendrogram representing the relation between marker classes. The colours used for the dendrogram labels are taken from the default colours (see `getStockcol`) so as to match the colours with other spatial proteomics visualisations such as `plot2D`.

### Usage

```r
mrkHClust(object, fcol = "markers", distargs, hclustargs, 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`.
- **distargs**: A list of arguments to be passed to the `dist` function.
- **hclustargs**: A list of arguments to be passed to the `hclust` function.
- **plot**: A logical defining whether the dendrogram should be plotted. Default is `TRUE`.
- **...**: Additional parameters passed when plotting the `dendrogram`.

### Value

Invisibly returns a matrix of average occupancy profiles for all marker classes defined in `fcol`. 

Author(s)
Laurent Gatto

Examples

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

---

**mrkVecToMat**  
*Create a marker vector or matrix.*

**Description**

Functions producing a new vector (matrix) marker vector set from an existing matrix (vector) marker set.

**Usage**

```r
mrkVecToMat(object, vfcol = "markers", mfcol = "Markers")
mrkMatToVec(object, mfcol = "Markers", vfcol = "markers")
mrkMatAndVec(object, vfcol = "markers", mfcol = "Markers")
showMrkMat(object, mfcol = "Markers")
isMrkMat(object, fcol = "Markers")
isMrkVec(object, fcol = "markers")
mrkEncoding(object, fcol = "markers")
```

**Arguments**

- `object`  
  An MSnSet object
- `vfcol`  
  The name of the vector marker feature variable. Default is "markers".
- `mfcol`  
  The name of the matrix marker feature variable. Default is "Markers".
- `fcol`  
  A marker feature variable name.

**Details**

Sub-cellular markers can be encoded in two different ways. Sets of spatial markers can be represented as character vectors (character or factor, to be accurate), stored as feature metadata, and proteins of unknown or uncertain localisation (unlabelled, to be classified) are marked with the "unknown" character. While very handy, this encoding suffers from some drawbacks, in particular the difficulty to label proteins that reside in multiple (possible or actual) localisations. The markers vector feature data is typically named markers. A new matrix encoding is also supported. Each spatial compartment is defined in a column in a binary markers matrix and the resident proteins are encoded with 1s. The markers matrix feature data is typically named Markers. If proteins are
assigned unique localisations only (i.e. no multi-localisation) or their localisation is unknown (un-
labelled), then both encodings are equivalent. When the markers are encoded as vectors, features of
unknown localisation are defined as \( fData(object)[, \text{fcol}] == "unknown" \). For matrix-encoded
markers, unlabelled proteins are defined as \( \text{rowSums}(fData(object)[, \text{fcol}]) == 0 \).

The \texttt{mrkMatToVec} and \texttt{mrkVecToMat} functions enable the conversion from matrix (vector) to vector
(matrix). The \texttt{mrkMatAndVec} function generates the missing encoding from the existing one. If
the destination encoding already exists, or, more accurately, if the feature variable of the destination
encoding exists, an error is thrown. During the conversion from matrix to vector, if multiple
possible label exists, they are dropped, i.e. they are converted to "unknown". Function \texttt{isMrkVec}
and \texttt{isMrkMat} can be used to test if a marker set is encoded as a vector or a matrix. \texttt{mrkEncoding}
returns either "vector" or "matrix" depending on the nature of the markers.

**Value**

An updated \texttt{MSnSet} with a new vector (matrix) marker set.

**Author(s)**

Laurent Gatto and Lisa Breckels

**See Also**

Other functions that operate on markers are \texttt{getMarkers}, \texttt{getMarkerClasses} and \texttt{markerMSnSet}. To add markers to an existing \texttt{MSnSet}, see the \texttt{addMarkers} function and \texttt{pRolocmarkers}, for a list of suggested markers.

**Examples**

```r
library("pRolocdata")
data(dunkley2006)
dunk <- mrkVecToMat(dunkley2006)
head(fData(dunk)$Markers)
fData(dunk)$markers <- NULL
dunk <- mrkMatToVec(dunk)
stopifnot(all.equal(fData(dunkley2006)$markers,
fData(dunk)$markers))
```

**nbClassification**

\[ nb \text{ classification} \]

**Description**

Classification using the naive Bayes algorithm.

**Usage**

\[ \text{nbClassification}(\text{object}, \text{assessRes}, \text{scores} = \text{c("prediction", "all", "none")}, \text{laplace}, \text{fcol} = \text{"markers"}, ...) \]
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

```r
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")
```

Description

Classification algorithm parameter for the naive Bayes algorithm.

Usage

```r
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"),
decimal, 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^(-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.
orderGoAnnotations

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.

Description

For a given matrix of annotation information, this function returns the information ordered according to the best fit with the data.

Usage

orderGoAnnotations(object, fcol = "GOAnnotations", k = 1:5, n = 5, p = 1/3, verbose = TRUE, seed)

Arguments

object An instance of class MSnSet.
fcol The name of the annotations matrix. Default is GOAnnotations.
k The number of clusters to test. Default is k = 1:5
n The minimum number of proteins per component cluster.
p The normalisation factor, per k tested
verbose A logical indicating if a progress bar should be displayed. Default is TRUE.
seed An optional random number generation seed.
Details

As there are typically many protein/annotation sets that may fit the data we order protein sets by best fit i.e. cluster tightness, by computing the mean normalised Euclidean distance for all instances per protein set.

For each protein set i.e. proteins that have been labelled with a specified term/information criteria, we find the best \( k \) cluster components for the set (the default is to test \( k = 1:5 \)) according to the minimum mean normalised pairwise Euclidean distance over all component clusters. (Note: when testing \( k \) if any components are found to have less than \( n \) proteins these components are not included and \( k \) is reduced by 1).

Each component cluster is normalised by \( N^p \) (where \( N \) is the total number of proteins per component, and \( p \) is the power). Hueristially, \( p = 1/3 \) and normalising by \( N^{1/3} \) has been found the optimum normalisation factor.

Candidates in the matrix are ordered according to lowest mean normalised pairwise Euclidean distance as we expect high density, tight clusters to have the smallest mean normalised distance. This function is a wrapper for running \texttt{clustDist}, \texttt{getNormDist}, see the "Annotating spatial proteomics data" vignette for more details.

Value

An updated \texttt{MSnSet} containing the newly ordered \texttt{fcol} matrix.

Author(s)

Lisa M Breckels

See Also

\texttt{addGoAnnotations} and example therein.

\begin{verbatim}
orgQuants
\end{verbatim}

\textbf{orgQuants} \hfill \textit{Returns organelle-specific quantile scores}

Description

This function produces organelle-specific quantiles corresponding to the given classification scores.

Usage

\begin{verbatim}
orgQuants(object, fcol, scol, mcol = "markers", t, verbose = TRUE)
\end{verbatim}

Arguments

\begin{verbatim}
object \hfill An instance of class \texttt{"MSnSet"}.
fcol \hfill The name of the prediction column in the \texttt{featureData} slot.
scol \hfill The name of the prediction score column in the \texttt{featureData} slot. If missing, created by pasting \texttt{\'scores\'} after \texttt{fcol}.
mcol \hfill The name of the column containing the training data in the \texttt{featureData} slot. Default is \texttt{markers}.
t \hfill The quantile threshold.
verbose \hfill If \texttt{TRUE}, the calculated thresholds are printed.
\end{verbatim}
perTurboClassification

Value

A named vector of organelle thresholds.

Author(s)

Lisa Breckels

See Also

getPredictions to get organelle predictions based on calculated thresholds.

Examples

library("pRolocdata")
data(dunkley2006)
res <- svmClassification(dunkley2006, fcol = "pd.markers",
  sigma = 0.1, cost = 0.5)
## 50% top predictions per class
ts <- orgQuants(res, fcol = "svm", t = .5)
getPredictions(res, fcol = "svm", t = ts)

perTurboClassification

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 \texttt{MSnSet} with \texttt{perTurbo} and \texttt{perTurbo.scores} feature variables storing the classification results and scores respectively.

Author(s)

Thomas Burger and Samuel Wieczorek

References


Examples

\begin{verbatim}
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")
\end{verbatim}

\section*{perTurboOptimisation \hspace{1cm} \textit{PerTurbo parameter optimisation}}

\section*{Description}

Classification parameter optimisation for the PerTurbo algorithm

\section*{Usage}

\begin{verbatim}
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)
\end{verbatim}
Arguments

- **object**
  An instance of class "MSnSet".

- **fcol**
  The feature meta-data containing marker definitions. Default is markers.

- **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.

- **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".

- **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

```r
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 discovery 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", "VEV", "VVV"), 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 original 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 specifying 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` argument 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


Examples

```r
## 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 or 3 dimensional feature distribution plots to illustrate localization clusters. Rows/features containing NA values are removed prior to dimension reduction. `plot3D` relies on the `rgl` package, that will be loaded automatically.
Usage

plot2D(object, fcol = "markers", fpch, unknown = "unknown", dims = 1:2,
      score = 1, method = "PCA", methargs, axsSwitch = FALSE,
mirrorX = FALSE, mirrorY = FALSE, col, pch, cex, index = FALSE,
      idx.cex = 0.75, addLegend, identify = FALSE, plot = TRUE, ...)

## S4 method for signature 'MSnSet'
plot2D(object, fcol = "markers", dims = c(1, 2, 3),
        radius1 = 0.1, radius2 = radius1 * 2, 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) desining the groups to be differentiated using different point symbols.

unknown  
A character (default is "unknown") defining how proteins of unknown/unlabelled localisation are labelled.

dims  
A numeric of length 2 (or 3 for plot3D) defining the dimensions to be plotted. Defaults are c(1,2) and c(1, 2, 3). 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  
A character describe how to transform the data or what to plot. One of "PCA" (default), "MDS", "kpca", "t-SNE" or "lda", defining what dimensionality reduction is applied: principal component analysis (see prcomp), classical multi-dimensional scaling (see cmdscale), kernel PCA (see kpca), t-SNE (see tsne) or linear discriminant analysis (see lda). The last method uses fcol to defined the sub-cellular clusters so that the ration between within ad between cluster variance is maximised. All the other methods are unsupervised and make use fcol only to annotate the plot. "scree" can also be used to produce a scree plot. "hexbin" applies PCA to the data and uses bivariate binning into hexagonal cells from hexbin to emphasise cluster density.

If none is used, the data is plotted as is, i.e. without any transformation. In this case, object can either be an MSnSet or a matrix (as invisibly returned by plot2D). This enables to re-generate the figure without computing the dimensionality reduction over and over again, which can be time consuming for certain methods. If object is a matrix, an MSnSet containing the feature meta-data must be provided inmethargs (see below for details).

Available methods are listed in plot2Dmethods.

methargs  
A list of arguments to be passed when method is called. If missing, the data will be scaled and centred prior to PCA. If method = "none" and object is a matrix, then the first and only argument of methargs must be an MSnSet with matching features with object.

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.
plot2D

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.

addLegend
A character indicating where to add the legend. See addLegend for details. If missing (default), no legend is added.

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.

radius1
A numeric specifying the radius of feature of unknown localisation. Default is 0.1, which is specified on the data scale. See plot3d for details.

radius2
A numeric specifying the radius of marker feature. Default is radius * 2.

Details

• Note that plot2D has been updated in version 1.3.6 to support more organelle classes than colours defined in getStockcol. In such cases, the default 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.

• Version 1.11.3: to plot data as is, i.e. without any transformation, method can be set to "none" (as opposed to passing pre-computed values to method as a matrix, in previous versions). If object is an MSnSet, the untransformed values in the assay data will be plotted. If object is a matrix with coordinates, then a matching MSnSet must be passed to methargs.

Value

Used for its side effects of generating a plot. Invisibly returns the 2 or 3 dimensions that are plotted.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

addLegend to add a legend to plot2D figures (the legend is added by default on plot3D) and plotDist for alternative graphical representation of quantitative organelle proteomics data. plot2Ds to overlay 2 data sets on the same PCA plot.

Examples

library("pRolocdata")
data(dunkley2006)
plot2D(dunkley2006, fcol = NULL)
plot2D(dunkley2006, fcol = NULL, col = "black")
plot2D(dunkley2006, fcol = "markers")
addLegend(dunkley2006,
plot2Ds

Draw 2 data sets on one PCA plot

Description

Takes 2 linkS4class(MSnSet) instances as input to plot the two data sets on the same PCA plot. The second data points are projected on the PC1 and PC2 dimensions calculated for the first data set.

Usage

plot2Ds(object, pcol, fcol = "markers", cex.x = 1, cex.y = 1,
pch.x = 21, pch.y = 23, col, mirrorX = FALSE, mirrorY = FALSE,
plot = TRUE, ...)

Arguments

object An MSnSet or a MSnSetList. In the latter case, only the two first elements of the list will be used for plotting and the others will be silently ignored.
If `object` is an `MSnSet`, a factor or the name of a phenotype variable (phenodata slot) defining how to split the single `MSnSet` into two or more data sets. Ignored if `object` is a `MSnSetList`.

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.

Character expansion for the first data set. Default is 1.

Character expansion for the second data set. Default is 1.

Plotting character for the first data set. Default is 21.

Plotting character for the second data set. Default is 23.

A vector of colours to highlight the different classes defined by `fcol`. If missing (default), default colours are used (see `getStockcol`).

A logical indicating whether the x axis should be mirrored?

A logical indicating whether the y axis should be mirrored?

If TRUE (default), a plot is produced.

... Additional parameters passed to `plot` and `points`.

Used for its side effects of producing a plot. Invisibly returns an object of class `plot2Ds`, which is a list with the PCA analyses results (see `prcomp`) of the first data set and the new coordinates of the second data sets, as used to produce the plot and the respective point colours. Each of these elements can be accessed with `data1`, `data2`, `col1` and `code2` respectively.

Laurent Gatto

See also `plot2D` to plot a single data set and `move2Ds` for an animation.

```r
library("pRolocdata")
data(tan2009r1)
data(tan2009r2)
msn1 <- MSnSetList(list(tan2009r1, tan2009r2))
plot2Ds(msn1)
## tweaking the parameters
plot2Ds(list(tan2009r1, tan2009r2),
        fcol = NULL, cex.x = 1.5)
## input is 1 MSnSet containing 2 data sets
data(dunkley2006)
plot2Ds(dunkley2006, pcol = "replicate")
## no plot, just the data
res <- plot2Ds(dunkley2006, pcol = "replicate",
                plot = FALSE)
res
head(data1(res))
head(col1(res))
```
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, type = "b", 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.
type        Character string defining the type of lines. For example "p" for points, "l" for 
lines, "b" for both. See plot for all possible types.
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 the data matrix.

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")
Classifier 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 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 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)
plot(params)
plotCount(params)
levelPlot(params)
getParams(params)
res <- plsdaClassification(dunkley2006, params)
getPredictions(res, fcol = "plsda")
getPredictions(res, fcol = "plsda", t = 0.9)
plot2D(res, fcol = "plsda")
Description
Classification parameter optimisation for the partial least square discriminant 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.
- 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.
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

See Also

`addMarkers` to add markers to an MSnSet and `markers` for more information about marker encoding.

Examples

```r
pRolocmarkers()
table(pRolocmarkers("atha"))
table(pRolocmarkers("hsap"))
```
QSep-class

Quantify resolution of a spatial proteomics experiment

Description

The QSep infrastructure provide a way to quantify the resolution of a spatial proteomics experiment, i.e. to quantify how well annotated sub-cellular clusters are separated from each other.

The QSep function calculates all between and within cluster average distances. These distances are then divided column-wise by the respective within cluster average distance. For example, for a dataset with only 2 spatial clusters, we would obtain

\[
\begin{array}{ccc}
  c_1 & d_1 & d_2 \\
  c_2 & d_2 & d_2
\end{array}
\]

Normalised distance represent the ratio of between to within average distances, i.e. how much bigger the average distance between cluster \( c_i \) and \( c_j \) is compared to the average distance within cluster \( c_i \).

\[
\begin{array}{ccc}
  c_1 & 1 & d_2 \\
  c_2 & \frac{d_2}{d_1} & 1
\end{array}
\]

Note that the normalised distance matrix is not symmetric anymore and the normalised distance ratios are proportional to the tightness of the reference cluster (along the columns).

Objects from the Class

Objects can be created by calls using the constructor QSep (see below).

Slots

- \texttt{x}: Object of class "matrix" containing the pairwise distance matrix, accessible with \texttt{qseq(. , norm = FALSE)}.
- \texttt{xnorm}: Object of class "matrix" containing the normalised pairwise distance matrix, accessible with \texttt{qsep(.,norm = TRUE)} or \texttt{qsep(.)}.
- \texttt{object}: Object of class "character" with the variable name of \texttt{MSnSet} object that was used to generate the QSep object.
- \texttt{__classVersion__}: Object of class "Versions" storing the class version of the object.

Extends

Class "Versioned", directly.

Methods and functions

- \texttt{QSeq signature(object = "MSnSet", fcol = "character")}: constructor for QSep objects. The \texttt{fcol} argument defines the name of the feature variable that annotates the sub-cellular clusters. Non-marker proteins, that are marked as "unknown" are automatically removed prior to distance calculation.
**QSep-class**

qsep signature(object = "QSep", norm = "logical"): accessor for the normalised (when norm is TRUE, which is default) and raw (when norm is FALSE) pairwise distance matrices.

names signature(object = "QSep"): method to retrieve the names of the sub-cellular clusters originally defined in QSep’s fcol argument. A replacement method names(.) <- is also available.

summary signature(object = "QSep", ..., verbose = "logical"): Invisible return all between cluster average distances and prints (when verbose is TRUE, default) a summary of those.

levelPlot signature(object = "QSep", norm = "logical"); plots an annotated heatmap of all normalised pairwise distances. norm (default is TRUE) defines whether normalised distances should be plotted. Additional arguments ... are passed to the levelplot.

plot signature(object = "QSep", norm = "logical"); produces a boxplot of all normalised pairwise distances. The red points represent the within average distance and black points between average distances. norm (default is TRUE) defines whether normalised distances should be plotted.

### Author(s)

Laurent Gatto <lg390@cam.ac.uk>

### Examples

```r
## Test data from Christoforou et al. 2016
library("pRolocdata")
data(hyperLOPIT2015)

## Create the object and get a summary
hlq <- QSep(hyperLOPIT2015)
hlq
summary(hlq)

## mean distance matrix
qsep(hlq, norm = FALSE)

## normalised average distance matrix
qsep(hlq)

## Update the organelle cluster names for better rendering on the plots
names(hlq) <- sub(“/”, “\n”, names(hlq))
names(hlq) <- sub(" - ", "\n", names(hlq))
names(hlq)

## Heatmap of the normalised intensities
levelPlot(hlq)

## Boxplot of the normalised intensities
par(mar = c(3, 10, 2, 1))
plot(hlq)

## Boxplot of all between cluster average distances
x <- summary(hlq, verbose = FALSE)
boxplot(x)
```
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)
getParam(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 (vector or matrix) 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. See markers for details about markers encoding.

Examples

library(pRolocdata)
data(tan2009r1)
dim(tan2009r1)
smp <- sampleMSnSet(tan2009r1, fcol = "markers")
dim(smp)
getMarkers(tan2009r1)
getMarkers(smp)
**setLisacol**

*Manage default colours and point characters*

**Description**

These functions allow to get/set the colours and point character that are used when plotting organelle clusters and unknown features. These values are parametrised at the session level. Two palettes are available: the default palette (previously *Lisa’s colours*) containing 30 colours and the old (original) palette, containing 13 colours.

**Usage**

```r
setLisacol()
getLisacol()
getOldcol()
setOldcol()
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**

The `set` functions set (and invisibly returns) colours. The `get` functions returns a character vector of colours. For the `pch` functions, numerics rather than characters.
### showGOEvidenceCodes

**Author(s)**
Laurent Gatto

**Examples**

```r
## 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()
getStockcol()
getOldcol()
```

---

<table>
<thead>
<tr>
<th>showGOEvidenceCodes</th>
<th>GO Evidence Codes</th>
</tr>
</thead>
</table>

**Description**

This function prints a textual description of the Gene Ontology evidence codes.

**Usage**

```r
showGOEvidenceCodes()
getGOEvidenceCodes()
```

**Value**

These functions are used for their side effects of printing evidence codes and their description.

**Author(s)**
Laurent Gatto

**Examples**

```r
showGOEvidenceCodes()
getGOEvidenceCodes()
```
Description

A class for spatial proteomics visualisation, that upon instantiation, pre-computes all defined visualisations. Objects can be created with the SpatProtVis constructor and visualised with the plot method.

The class is essentially a wrapper around several calls to plot2D that stores the dimensionality reduction outputs, and is likely to be updated in the future.

Usage

SpatProtVis(x, methods, dims, methargs, ...)

Arguments

x
   An instance of class MSnSet to visualise.

methods
   Dimensionality reduction methods to be used to visualise the data. Must be contained in plot2Dmethods (except "scree"). See plot2D for details.

dims
   A list of numerics defining dimensions used for plotting. Default are 1 and 2. If provided, the length of this list must be identical to the length of methods.

methargs
   A list of additional arguments to be passed for each visualisation method. If provided, the length of this list must be identical to the length of methods.

... Additional arguments. Currently ignored.

Slots

vismats: A "list" of matrices containing the feature projections in 2 dimensions.

data: The original spatial proteomics data stored as an "MSnSet".

methargs: A "list" of additional plotting arguments.

objname: A "character" defining how to name the dataset. By default, this is set using the variable name used at object creation.

Methods

plot: Generates the figures for the respective methods and additional arguments defined in the constructor. If used in an interactive session, the user is prompted to press 'Return' before new figures are displayed.

show: A simple textual summary of the object.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

The data for the individual visualisations is created by plot2D.
**Examples**

```r
library("pRolocdata")
data(dunkley2006)
## Default parameters for a set of methods
## (in the interest of time, don't use t-SNE)
m <- c("PCA", "MDS", "kpca")
vis <- SpatProtVis(dunkley2006, methods = m)
vis
plot(vis)
plot(vis, legend = "topleft")

## Setting method arguments
margs <- c(list(kpar = list(sigma = 0.1)),
           list(kpar = list(sigma = 1.0)),
           list(kpar = list(sigma = 10)),
           list(kpar = list(sigma = 100)))
vis <- SpatProtVis(dunkley2006,
                   methods = rep("kpca", 4),
                   methargs = margs)
par(mfrow = c(2, 2))
plot(vis)

## Multiple PCA plots but different PCs
dims <- list(c(1, 2), c(3, 4))
vis <- SpatProtVis(dunkley2006, methods = c("PCA", "PCA"),
                   dims = dims)
plot(vis)
```

---

**subsetMarkers**

Subsets markers

**Description**

Subsets a matrix of markers by specific terms

**Usage**

```r
subsetMarkers(object, fcol = "GOAnnotations", keep)
```

**Arguments**

- `object` An instance of class MSnSet.
- `fcol` The name of the markers matrix. Default is GOAnnotations.
- `keep` Integer or character vector specifying the columns to keep in the markers matrix, as defined by fcol.

**Value**

An updated MSnSet

**Author(s)**

Lisa M Breckels
svmClassification

See Also

addGoAnnotations and example therein.

---

svmClassification  

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

```r
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)
getParam(params)
res <- svmClassification(dunkley2006, params)
getPredictions(res, fcol = "svm")
getPredictions(res, fcol = "svm", t = 0.75)
plot2D(res, fcol = "svm")
```
Description

Classification parameter optimisation for the support vector machine algorithm.

Usage

```r
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 thrown, 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 \texttt{minMarkers}) as the parameter optimisation is likely to fail or, at least, produce unreliable results for that class.

Value

If successful, the test invisibly returns \texttt{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

```r
library("pRolocdata")
data(dunkley2006)
getMarkers(dunkley2006)
testMarkers(dunkley2006)
toosmall <- testMarkers(dunkley2006, xval = 15)
toosmall
try(testMarkers(dunkley2006, xval = 15, error = TRUE))
```
Create a stratified 'test' MSnSet

Description

This function creates a stratified 'test' MSnSet which can be used for algorithmic 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)))
### thetas

**Draw matrix of thetas to test**

**Description**

The possible weights to be considered is a sequence from 0 (favour auxiliary data) to 1 (favour primary data). Each possible combination of weights for nclass classes must be tested. The `thetas` function produces a weight matrix for nclass columns (one for each class) with all possible weight combinations (number of rows).

**Usage**

```r
thetas(nclass, by = 0.5, length.out, verbose = TRUE)
```

**Arguments**

- `nclass` Number of marker classes
- `by` The increment of the weights. One of 1, 0.5, 0.25, 2, 0.1 or 0.05.
- `length.out` The desired length of the weight sequence.
- `verbose` A logical indicating if the weight sequences should be printed out. Default is TRUE.

**Value**

A matrix with all possible theta weight combinations.

**Author(s)**

Lisa Breckels

**Examples**

```r
dim(thetas(4, by = 0.5))
dim(thetas(4, by = 0.2))
dim(thetas(5, by = 0.2))
dim(thetas(5, length.out = 5))
dim(thetas(6, by = 0.2))
```

---

**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>
zerosInBinMSnSet  

**Compute the number of non-zero values in each marker classes**

**Description**

The function assumes that its input is a binary MSnSet and computes, for each marker class, the number of non-zero expression profiles. The function is meant to be used to produce heatmaps (see the example) and visualise binary (such as GO) MSnSet objects and assess their utility: all zero features/classes will not be informative at all (and can be filtered out with `filterBinMSnSet`) while features/classes with many annotations (GO terms) are likely not be be informative either.

**Usage**

```r
zerosInBinMSnSet(object, fcol = "markers", as.matrix = TRUE, percent = TRUE)
```

**Arguments**

- `object`: An instance of class MSnSet with binary data.
- `fcol`: A character defining the feature data variable to be used as markers. Default is "markers".
- `as.matrix`: If TRUE (default) the data is formatted and returned as a matrix. Otherwise, a list is returned.
- `percent`: If TRUE, percentages are returned. Otherwise, absolute values.

**Value**

A matrix or a list indicating the number of non-zero value per marker class.

**Author(s)**

Laurent Gatto

**See Also**

- `filterBinMSnSet`

**Examples**

```r
count <- zerosInBinMSnSet(hyperLOPIT2015goCC)
count <- zerosInBinMSnSet(hyperLOPIT2015goCC, percent = FALSE)
pal <- colorRampPalette(c("white", "blue"))
levelplot(count, xlab = "Number of non-0s", ylab = "Marker class", col.regions = pal(140))
```
Index

*Topic classes
  AnnotationParams-class, 8
  ClustDist-class, 14
  ClustDistList-class, 15
  GenRegRes-class, 22
  QSep-class, 65
  SpatProtVis-class, 72

*Topic methods
  chi2-methods, 11
  exprsToRatios-methods, 18
  MLearn-methods, 41
  nndist-methods, 47

*Topic package
  pRoloc-package, 3
  [,ClustDistList,ANY,ANY,ANY-method (ClustDistList-class), 15
  [,ClustDistList,ANY,missing,missing-method (ClustDistList-class), 15
  [,MartInstanceList,ANY,ANY,ANY-method (MartInstance-class), 40
  [,MartInstanceList,ANY,ANY-method (MartInstance-class), 40
  [,MartInstanceList-method (MartInstance-class), 40
  [[,ClustDistList,ANY,ANY-method (ClustDistList-class), 15
  [[,ClustDistList,ANY,missing-method (ClustDistList-class), 15
  [[,MartInstanceList,ANY,ANY-method (MartInstance-class), 40
  [[,MartInstanceList-method (MartInstance-class), 40
  addGoAnnotations, 4
  addLegend, 6, 58
  addMarkers, 6, 38, 45, 64
  andy2011params (AnnotationParams-class), 8
  AnnotationParams, 24, 38
  AnnotationParams (AnnotationParams-class), 8
  AnnotationParams-class, 8
  as.data.frame.MartInstance (MartInstance-class), 40
  as.data.frame.MartInstanceList (MartInstance-class), 40
  checkFeatureNamesOverlap, 9
  checkVarOverlap, 10
  chi2, 17
  chi2(chi2-methods), 11
  chi2,matrix,matrix-method (chi2-methods), 11
  chi2,matrix,numeric-method (chi2-methods), 11
  chi2,numeric,matrix-method (chi2-methods), 11
  chi2,numeric,numeric-method (chi2-methods), 11
  choc2-methods, 11
  chol2inv, 52, 54
  class::QSep (QSep-class), 65
  class:AnnotationParams (AnnotationParams-class), 8
  class:ClustDist (ClustDist-class), 14
  class:ClustDistList (ClustDistList-class), 15
  class:GenRegRes (GenRegRes-class), 22
  class:SpatProtVis (SpatProtVis-class), 72
  class:ThetaRegRes (GenRegRes-class), 22
  classWeights, 12
  ClustDist, 13–15, 26
  ClustDist (ClustDist-class), 14
  clustDist, 13
  ClustDist-class, 14
  ClustDistList, 13, 14, 26
  ClustDistList (ClustDistList-class), 15
  ClustDistList-class, 15
  cmdscale, 57
  col1 (plot2Ds), 59
  col2 (plot2Ds), 59
  combineThetaRegRes (GenRegRes-class), 22
  data1 (plot2Ds), 59
  data2 (plot2Ds), 59
dendrogram, 43
dist, 43
exprsToRatios(exprsToRatios-method)
empPvalues, 11, 17
estimateMasterFdr, 37
eexprsToRatios(exprsToRatios-methods, 18
exprsToRatios,MSnSet-method
(exprsToRatios-methods, 18
exprsToRatios-methods, 18
f1Count (GenRegRes-class, 22
f1Count,GenRegRes-method
(GenRegRes-class, 22
f1Count,ThetaRegRes-method
(GenRegRes-class, 22
favourPrimary(GenRegRes-class, 22
fDataToUnknown, 18
FeaturesOfInterest, 29
filterAttrs(MartInstance-class, 40
filterBinMSnSet, 19, 79
filterMaxMarkers, 20
filterMinMarkers, 21
filterNA, 56
filterZeroCols, 19, 21
filterZeroRows, 19
filterZeroRows(filterZeroCols, 21
flipGoTermId(goIdToTerm), 28
GenRegRes, 30, 32, 35, 36, 46–48, 50, 52, 54, 62, 63, 67, 68, 74, 75
GenRegRes(GenRegRes-class, 22
GenRegRes-class, 22
getAnnotationParams, 5
getAnnotationParams
(AnnotationParams-class, 8
getF1Scores (GenRegRes-class, 22
getF1Scores,GenRegRes-method
(GenRegRes-class, 22
getF1Scores,ThetaRegRes-method
(GenRegRes-class, 22
getFilterList(MartInstance-class, 40
getGOEvidenceCodes (showGOEvidenceCodes), 71
getGOFromFeatures, 8, 9, 23
get Lisacol(setLisacol), 70
getMarkerClasses, 24, 26, 45
getMarkers, 25, 25, 45, 76
getMartInstanceList
(MartInstance-class, 40
getMartTab(MartInstance-class, 40
getNormDist, 26
getOldcol(setLisacol), 70
getParams (GenRegRes-class, 22
getParams,ClustRegRes-method
(undocumented), 78
getParams,GenRegRes-method
(GenRegRes-class, 22
getParams,ThetaRegRes-method
(GenRegRes-class, 22
getPredictions, 27, 41, 52
getRegularisedParams (GenRegRes-class, 22
getRegularisedParams,GenRegRes-method
(GenRegRes-class, 22
getRegularisedParams,GenRegRes-method
(GenRegRes-class, 22
gin v, 52, 54
goIdToTerm, 28
goTermTold (goIdToTerm), 28
hclust, 43
hexbin, 57
highlightOnPlot, 29
highlightOnPlot3D (highlightOnPlot), 29
identify, 58
isMrkMat (mrkVecToMat), 44
isMrkVec (mrkVecToMat), 44
knn, 30, 31
knnClassification, 30, 32
knnOptimisation, 22, 30, 31
knnOptimization (knnOptimisation), 31
knnPrediction (knnClassification), 30
knnRegularisation (knnOptimisation), 31
knt1Classification, 32, 35
knt1Optimisation, 22, 33, 33
kpca, 57
ksvm, 35, 36
ksvmClassification, 35, 36
ksvmOptimisation, 35, 36
INDEX

ksvmOptimization (ksvmOptimisation), 36
ksvmPrediction (ksvmClassification), 35
ksvmRegularisation (ksvmOptimisation), 36
lapply, ClustDistList-method
(ClustDistList-class), 15
lapply, MartInstanceList, ANY-method
(MartInstance-class), 40
lapply, MartInstanceList-method
(MartInstance-class), 40
lda, 57
legend, 6
length, ClustDistList-method
(ClustDistList-class), 15
levelPlot (GenRegRes-class), 22
levelplot, 66
levelPlot, ClustRegRes-method
(undocumented), 78
levelPlot, GenRegRes-method
(GenRegRes-class), 22
levelPlot, QSep-method (QSep-class), 65
lopims, 37
lopims1 (lopims), 37
lopims2 (lopims), 37
lopims3 (lopims), 37
lopims4 (lopims), 37
lopims5 (lopims), 37
makeGoSet, 8, 9, 38
makeMaster, 37
markerMSnSet, 39, 45, 69, 77
markers, 7, 25, 26, 39, 64, 69
markers (mrkVecToMat), 44
MartInstance (MartInstance-class), 40
MartInstance-class, 40
MartInstanceList (MartInstance-class), 40
MartInstanceList-class
(MartInstance-class), 40
minMarkers, 40, 76
MLearn, 41
MLearn, formula, MSnSet, clusteringSchema, missing-method
(MLearn-methods), 41
MLearn, formula, MSnSet, learnerSchema, numeric-method
(MLearn-methods), 41
MLearn, formula, MSnSet, learnerSchema, xvalSpec-method
(MLearn-methods), 41
MLearn-methods, 41
MLearnMSnSet (MLearn-methods), 41
move2Ds, 42, 60
mrkEncoding (mrkVecToMat), 44
mrkHClust, 43
mrkMatAndVec (mrkVecToMat), 44
mrkMatToVec (mrkVecToMat), 44
mrkVecToMat, 44
MSnbase, 4
MSnSetList, 42, 60
MSnSetMLearn (MLearn-methods), 41
naiveBayes, 46, 47
names, ClustDistList-method
(ClustDistList-class), 15
names, QSep-method (QSep-class), 65
names <-, ClustDistList, ANY-method
(ClustDistList-class), 15
names <-, QSep, character-method
(QSep-class), 65
nbClassification, 45, 47
nbOptimisation, 46, 46
nbOptimization (nbOptimisation), 46
nbPrediction (nbClassification), 45
nbRegularisation (nbOptimisation), 46
nDatasets (MartInstance-class), 40
nndist (nndist-methods), 47
nndist, matrix, matrix-method
(nndist-methods), 47
nndist, matrix, missing-method
(nndist-methods), 47
nndist, MSnSet, missing-method
(nndist-methods), 47
nndist-methods, 47
nnet, 48, 49
nnetClassification, 48, 50
nnetOptimisation, 48, 49
nnetOptimization (nnetOptimisation), 49
nnetPrediction (nnetClassification), 48
nnetRegularisation (nnetOptimisation), 49
orderGoAnnotations, 50
orgQuants, 51
perTurboClassification, 52, 54
perTurboOptimisation, 52, 53
perTurboOptimization
(perTurboOptimisation), 53
phenoDisco, 55
plot, 61
plot, ClustDist, MSnSet-method
(ClustDist-class), 14
INDEX

plot, ClustDistList, missing-method (ClustDistList-class), 15
plot, ClustRegRes, missing-method (ClustDistList-class), 15
plot, GenRegRes, missing-method (GenRegRes-class), 22
plot, QSep, missing-method (QSep-class), 65
plot, QSep-method (QSep-class), 65
plot, SpatProtVis, missing-method (SpatProtVis-class), 72
plot, ThetaRegRes, missing-method (ThetaRegRes-class), 22
plot2D, 6, 43, 56, 60, 72
plot2Dmethods, 72
plot2Dmethods (plot2D), 56
plot2D, 42, 58, 59
plot3d, 58
plot3D, MsnSet-method (plot2D), 56
plotDist, 58, 61
plsd, 62
plsd2Classification, 62, 63
plsd2Optimisation, 22, 62, 63
plsd2Optimisation (plsdOptimisation), 63
plsd2Prediction (plsd2Classification), 62
plsdRegularisation (plsdOptimisation), 63
prcomp, 57, 60
prettyGoTermId (goIdToTerm), 28
pRoloc, pRoloc-package, 3
pRoloc-package, 3
pRolocmarkers, 6, 7, 45, 64
QSep (QSep-class), 65
qsep (QSep-class), 65
QSep-class, 65
randomForest, 67, 68
rfClassification, 67, 68
rfOptimisation, 67, 68
rfOptimisation (rfOptimisation), 68
rfPrediction (rfClassification), 67
rfRegularisation (rfOptimisation), 68
sampleMSnSet, 39, 69, 77
sapply, ClustDistList-method (ClustDistList-class), 15
sapply, MartInstanceList, ANY-method (MartInstance-class), 40
sapply, MartInstanceList-method (MartInstance-class), 40
setAnnotationParams (AnnotationParams-class), 8
setLisacol, 70
setOldcol (setLisacol), 70
setStockcol (setLisacol), 70
setStockpch (setLisacol), 70
setUnknowncol (setLisacol), 70
setUnknownpch (setLisacol), 70
show, AnnotationParams-method (AnnotationParams-class), 8
show, ClustDist-method (ClustDist-class), 14
show, ClustDistList-method (ClustDistList-class), 15
show, ClustRegRes-method (ClustDistList-class), 22
show, GenRegRes-method (GenRegRes-class), 22
show, MartInstance-method (MartInstance-class), 40
show, QSep-method (QSep-class), 65
show, SpatProtVis-method (SpatProtVis-class), 72
show, ThetaRegRes-method (ThetaRegRes-class), 22
showGOEvidenceCodes, 71
showMrkMat (mrkVecToMat), 44
solve, 52, 54
SpatProtVis (SpatProtVis-class), 72
SpatProtVis-class, 72
sub, 18
subsetMarkers, 73
summary, QSep-method (QSep-class), 65
svd, 52, 54
svm, 74, 75
svmClassification, 12, 74, 75
svmOptimisation, 12, 22, 74, 75
svmOptimisation (svmOptimisation), 75
svmPrediction (svmClassification), 74
svmRegularisation, 52
svmRegularisation (svmOptimisation), 75
Synapater, 37
synapater, 37
synergise, 37
testMarkers, 76
testMSnSet, 39, 69, 77
ThetaRegRes (ThetaRegRes-class), 22
ThetaRegRes-class (ThetaRegRes-class), 22
theta, 34, 78
tSNE, 57
undocumented, 78
unknownMSnSet, 69, 77
unknownMSnSet (markerMSnSet), 39

Versioned, 65
xvalSpec, 41

zerosInBinMSnSet, 19, 79