Package 'MSnbase'

April 22, 2016

Title Base Functions and Classes for MS-based Proteomics

Version 1.18.1

Description Basic plotting, data manipulation and processing
of MS-based Proteomics data.

Author Laurent Gatto <lg390@cam.ac.uk> with contributions from
Guangchuang Yu, Samuel Wieczorek, Vasile-Cosmin Lazar,
Vladislav Petyuk, Thomas Naake and Sebastian Gibb.

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

Depends R (>= 3.1), methods, BiocGenerics (>= 0.7.1), Biobase (>=
2.15.2), mzR, BiocParallel, ProtGenerics

Imports plyr, IRanges, preprocessCore, vsn, grid, reshape2, stats4,
affy, impute, pcaMethods, mzID (>= 1.5.2), MALDIquant (>=
1.12), digest, lattice, ggplot2, S4Vectors, Rcpp

Suggests testthat, zoo, knitr (>= 1.1.0), rols, Rdisop, pRoloc,
pRolocdata (>= 1.7.1), msdata, roxygen2, rgl, BiocStyle,
imputeLCMD, norm, gplots

LinkingTo Rcpp

License Artistic-2.0

LazyData yes

VignetteBuilder knitr

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

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

biocViews Infrastructure, Proteomics, MassSpectrometry,
QualityControl, DataImport

NeedsCompilation yes

R topics documented:

1

MSnbase-package   *MSnbase: Base Functions and Classes for MS-based Proteomics*

Description

MSnbase provides classes, methods and functions for visualisation, manipulation and processing of mass spectrometry data.

Important class are *"MSnExp"* (raw data file), *"MSnSet"* (quantitation data) and *"ReporterIons"* (reporter ions for labelled proteomics).

Other classes are *"Spectrum"* and the subclasses *"Spectrum1"* (for MS spectra) and *"Spectrum2"* (for MSMS spectra), *"MIAPE"* (Minimum Information about Proteomics Experiments) and *"MSnProcess"* (for processing information). These should however not be of direct utility to users.

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

Author(s)

Laurent Gatto

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

See the DESCRIPTION file for a complete list of contributors.

References

Laurent Gatto and Kathryn S. Lilley, MSnbase - an R/Bioconductor package for isobaric tagged mass spectrometry data visualization, processing and quantitation, Bioinformatics 28(2), 288-289 (2012).

Gatto L. and Lilley K.S., Towards reproducible MSMS data preprocessing, quality control and quantification. BSPR/EBI Proteomics Meeting, Hinxton, United Kingdom, 13-15 July 2010, http://dx.doi.org/10.1038/npre.2010.5010.1.

See Also

Introductory information, use cases and details are available from the vignettes:

- The demo vignette describe an use-case using a dummy data set provided with the package. It can be accessed with vignette("MSnbase-demo", package = "MSnbase").
- The development vignette describes the classes implemented in MSnbase and can be accessed with vignette("MSnbase-development", package = "MSnbase").
- Details about input/outcput capabilities and formats can be found in vignette("MSnbase-io", package = "MSnbase"

Complete listing of available documentation with library(help = "MSnbase").

---

addIdentificationData-methods
                    Adds Identification Data

Description

This methods add identification data to an experiment "MSnExp" or to a "MSnSet".

Details

The featureData slots in an "MSnExp" or an "MSnSet" instance provides only one row per MS2 spectrum but the identification is not always bijective. If multiple possible matches are present only the highest ranked identification is added.

The column nprot contains the number of members in the protein group; the columns accession and description contain a semicolon separated list of all matches sorted by their rank values. The columns npsm.prot and npep.prot represent the number of PSMs and peptides that were matched to a particular protein group. The column npsm.pep indicates how many PSMs were attributed to a peptide (as defined by its sequence pepseq).

Methods

signature(object = "MSnExp", id = "character", fcol = "character", icol = "character", verbose = "l
    Adds the identification data stored in mzIdentML files to a "MSnExp" instance. The method
    handles one or multiple mzIdentML files provided via id. id has to be a character vector
    of valid filenames. fcol and icol specify the columns in the featureData slot and the iden-
    tification data.frame that are used to merge both data together as character vectors (defaults
    are fcol = c("spectrum.file", "acquisition.number") and icol = c("spectrumFile", "acquisitionnum"))
    The verbose argument (default is TRUE) defines whether status messages should be showed.

signature(object = "MSnExp", id = "mzID", fcol = "character", icol = "character", verbose = "logica
Same as above but id could be an mzID object.

signature(object = "MSnExp", id = "mzIDCollection", fcol = "character", icol = "character", verbose
Same as above but id could be an mzIDCollection object.

signature(object = "MSnExp", id = "data.frame", fcol = "character", icol = "character", verbose = "
Same as above but id could be a data.frame.

signature(object = "MSnSet", id = "character", verbose = "logical") Adds the iden-
tification data stored in mzIdentML files to an ["MSnSet"](#) instance. The method handles one or
multiple mzIdentML files provided via id. id has to be a character vector of valid filenames.

fcol and icol specify the columns in the featureData slot and the identification data.frame
that are used to merge both data together as character vectors (defaults are fcol = c("spectrum.file", "acquisiti
and icol = c("spectrumFile", "acquisitionnum")).

The verbose argument (default is TRUE) defines whether status messages should be showed.

signature(object = "MSnSet", id = "mzID", fcol = "character", icol = "character", verbose = "logica
Same as above but id could be an mzID object.

signature(object = "MSnSet", id = "mzIDCollection", fcol = "character", icol = "character", verbose
Same as above but id could be an mzIDCollection object.

signature(object = "MSnSet", id = "data.frame", fcol = "character", icol = "character", verbose = "
Same as above but id could be a data.frame.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

See Also

[MSnExp](#) and [MSnSet](#).

Examples

```
## find path to a mzXML file
quantFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "mzXML$")
## find path to a mzIdentML file
identFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "dummyiTRAQ.mzid")

## create basic MSnExp
msexp <- readMSData(quantFile)

## add identification information
msexp <- addIdentificationData(msexp, identFile)

## access featureData; please note the multiple identification data
## for spectrum 1 (row 1)
fData(msexp)

idSummary(msexp)
```

---

averageMSnSet   *Generate an average* MSnSet

Description

Given a list of `MSnSet` instances, typically representing replicated experiments, the function returns an average `MSnSet`.

Usage

```
averageMSnSet(x, avg = function(x) mean(x, na.rm = TRUE), disp = npcv)
```

Arguments

x	A `list` of valid `MSnSet` instances to be averaged.
avg	The averaging function. Default is hte median after removing missing valiues, as computed by `function(x) median(x, na.rm = TRUE)`.
disp	The disperion function. Default is an non-parametric coefficient of variation that replaces the standard deviation by the median absolute deviation as computed by `mad(x)/abs(mean(x))`. See [npcv](#) for details. Note that the mad of a single value is 0 (as opposed to `NA` for the standard deviation, see example below).

Details

This function is aimed at facilitating the visualisation of replicated experiments and should not be used as a replacement for a statistical analysis.

The samples of the instances to be averaged must be identical but can be in a different order (they will be reordered by default). The features names of the result will correspond to the union of the feature names of the input `MSnSet` instances. Each average value will be computed by the `avg` function and the dispersion of the replicated measurements will be estimated by the `disp` function. These dispersions will be stored as a `data.frame` in the feature metadata that can be accessed with `fData(.)$disp`. Similarly, the number of missing values that were present when average (and dispersion) were computed are available in `fData(.)$disp`.

Currently, the feature metadata of the returned object corresponds the the feature metadata of the first object in the list (augmented with the missing value and dispersion values); the metadata of the features that were missing in this first input are missing (i.e. populated with NAs). This may change in the future.

Value

A new average `MSnSet`.

Author(s)

Laurent Gatto

See Also

[compfnames](#) to compare MSnSet feature names.

Examples

```
library("pRolocdata")
## 3 replicates from Tan et al. 2009
data(tan2009r1)
data(tan2009r2)
data(tan2009r3)
x <- MSnSetList(list(tan2009r1, tan2009r2, tan2009r3))
avg <- averageMSnSet(x)
dim(avg)
head(exprs(avg))
head(fData(avg)$nNA)
head(fData(avg)$disp)
## using the standard deviation as measure of dispersion
avg2 <-averageMSnSet(x, disp = sd)
head(fData(avg2)$disp)
## keep only complete observations, i.e proteins
## that had 0 missing values for all samples
sel <- apply(fData(avg)$nNA, 1 , function(x) all(x == 0))
avg <- avg[sel, ]
disp <- rowMax(fData(avg)$disp)
library("pRoloc")
setStockcol(paste0(getStockcol(), "AA"))
plot2D(avg, cex = 7.7 * disp)
title(main = paste("Dispersion: non-parametric CV",
                    paste(round(range(disp), 3), collapse = " - ")))
```

---

bin-methods   *Bin 'MSnExp' or 'Spectrum' instances*

Description

This method aggregates individual spectra (Spectrum instances) or whole experiments (MSnExp instances) into discrete bins. All intensity values which belong to the same bin are summed together.

Methods

signature(object = "MSnExp", binSize = "numeric", verbose = "logical") Bins all spectra in an MSnExp object. Use binSize to control the size of a bin (in Dalton, default is 1). Displays a control bar if verbose set to TRUE (default). Returns a binned MSnExp instance.

signature(object = "Spectrum", binSize = "numeric", breaks = "numeric") Bin the Spectrum object. Use binSize to control the size of a bin (in Dalton, default is 1). Similar to [hist](#) you could use breaks to specify the breakpoints between m/z bins. Returns a binned Spectrum instance.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

See Also

[clean](), [pickPeaks](), [smooth](), [removePeaks]() and [trimMz]() for other spectra processing methods.

Examples

```
s <- new("Spectrum2", mz=1:10, intensity=1:10)
intensity(s)
intensity(bin(s, binSize=2))

data(itraqdata)
sum(peaksCount(itraqdata))
itraqdata2 <- bin(itraqdata, binSize=2)
sum(peaksCount(itraqdata2))
processingData(itraqdata2)
```

---

calculateFragments-methods
                                          Calculate ions produced by fragmentation.

Description

These method calculates a-, b-, c-, x-, y- and z-ions produced by fragmentation.

Arguments

sequence	character, peptide sequence.
object	Object of class "[Spectrum2]()" or "missing" .
tolerance	numeric tolerance between the theoretical and measured MZ values (only available if object is not missing).
method	method used for for duplicated matches. Choose "highest" or "closest" to select the peak with the highest intensity respectively the closest MZ in the tolerance range. If "all" is given all possible matches in the tolerance range are reported (only available if object is not missing).
type	character vector of target ions; possible values: c("a", "b", "c", "x", "y", "z"); default: type=c("b", "y").
z	numeric desired charge state; default z=1.
modifications	named numeric vector of used modifications. The name must correspond to the one-letter-code of the modified amino acid and the numeric value must represent the mass that should be added to the original amino accid mass, default: Carbamidomethyl modifications=c(C=57.02146). Use Nterm or Cterm as names for modifications that should be added to the amino respectively carboxyl-terminus.

neutralLoss list, it has to have two named elments, namely water and ammonia that contain a character vector which type of neutral loss should be calculated. Currently neutral loss on the C terminal "Cterm", at the amino acids c("D", "E", "S", "T") for "water" (shown with an _) and c("K", "N", "Q", "R") for "ammonia" (shown with an *) are supported.

There is a helper function defaultNeutralLoss that returns the correct list. It has two arguments disableWaterLoss and disableAmmoniaLoss to remove single neutral loss options. See the example section for use cases.

verbose logical if TRUE (default) the used modifications are printed.

Methods

signature(sequence = "character", object = "missing", ...) Calculates the theoretical fragments for a peptide sequence. Returns a data.frame with the columns c("mz", "ion", "type",   "pos",

signature(sequence = "character", object = "Spectrum2", ...) Calculates and matches the theoretical fragments for a peptide sequence and a "Spectrum2" object. The ... arguments are passed to the internal functions. Currently tolerance, method and relative are supported. You could change the tolerance (default 25e-6) and decide whether this tolerance should be applied relative (default relative = TRUE) or absolute (relative = FALSE) to match the theoretical fragment MZ with the MZ of the spectrum. In cases of multiple matches use method to select the peak with the highest intensity (method = "highest", default) respectively closest MZ (method = "closes"). If method = "all" is set all possible matches in the current tolerance range are reported. Returns the same data.frame as above but the mz column represents the matched MZ values of the spectrum. Additionally there is a column error that contains the difference between the observed MZ (from the spectrum) to the theoretical fragment MZ.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

Examples

```
## find path to a mzXML file
file <- dir(system.file(package = "MSnbase", dir = "extdata"),
            full.name = TRUE, pattern = "mzXML$")

## create basic MSnExp
msexp <- readMSData(file)

## centroid them
msexp <- pickPeaks(msexp)

## calculate fragments for ACE with default modification
calculateFragments("ACE", modifications=c(C=57.02146))

## calculate fragments for ACE with an addition N-terminal modification
calculateFragments("ACE", modifications=c(C=57.02146, Nterm=229.1629))

## calculate fragments for ACE without any modifications
```

```
calculateFragments("ACE", modifications=NULL)

calculateFragments("VESITARHGEVLQLRPK",
                    type=c("a", "b", "c", "x", "y", "z"),
                    z=1:2)

calculateFragments("VESITARHGEVLQLRPK", msexp[[1]])

## neutral loss
defaultNeutralLoss()

## disable water loss on the C terminal
defaultNeutralLoss(disableWaterLoss="Cterm")

## real example
calculateFragments("PQR")
calculateFragments("PQR",
                    neutralLoss=defaultNeutralLoss(disableWaterLoss="Cterm"))
calculateFragments("PQR",
                    neutralLoss=defaultNeutralLoss(disableAmmoniaLoss="Q"))

## disable neutral loss completely
calculateFragments("PQR", neutralLoss=NULL)
```

---

chromatogram-methods   *Plots a chromatogram*

Description

The method plot the chromatogram for various types in inputs (see below). Additional arguments are

y One of "tic" (default) or "bpi" to plot the total ion current of base peak intensity chromatogram.

f Optional and only when the input is a data.frame. Otherwise, it is extracted automatically from object. f is used to print the filename on the figure.

legend A logical defining if the figure should be annotated.

plot A logical defining if the plot should be rendered.

ms A numeric defining what MS level spectra to use. Default is 1L.

... Additional arguments passed to the plot function.

xcms::plotChrom provides a similar functionality.

Value

The methods invisibly return the data.frame with the retention times (rt column) and intensities (either tic or bpi) used to generate the figure.

Methods

signature(object = "character") Plots the chromatogram for the mass-spectrometry data stored
in the object file. The file format must be support by mzR. See mzR::openMSfile for details.

signature(object = "mzRramp") Plots the chromatogram for the mzRramp instance. See the mzR
package for details.

signature(object = "data.frame") Plots the chromatogram using the relevant columns from
the data.frame instance, i.e retentionTime and totIonCurrent (for tic) and basePeakIntensity
(for bpi). Such a data.frame would typically be generated by extracting the header from an
mzRramp instance. See mzR::header for details.

Examples

```
f <- system.file("lockmass/LockMass_test.mzXML", package = "msdata")
x <- chromatogram(f, main = "Source: mzXML file")
head(x)
dim(x)
x <- chromatogram(f, main = "Source: mzXML file",
                  ylim = c(0, 100))
## Not run:
    library("mzR")
    ms <- openMSfile(f)
    chromatogram(ms, main = "Source: mzRramp",
                col = "red")
    hd <- header(ms)
    chromatogram(hd, main = "Source: mzRramp header",
                lty = "dashed")

    library("RforProteomics")
    f <- getPXD000001mzXML()
    chromatogram(f)
    grid()

## End(Not run)
```

---

clean-methods   *Cleans 'MSnExp' or 'Spectrum' instances*

Description

This method cleans out individual spectra (Spectrum instances) or whole experiments (MSnExp
instances) of 0-intensity peaks. Unless all is set to FALSE, origninal 0-intensity values are retained
only around peaks. If more than two 0's were separating two peaks, only the first and last ones,
those directly adjacent to the peak ranges are kept. If two peaks are separated by only one 0-
intensity value, it is retained. An illustrative example is shown below.

Methods

signature(object = "MSnExp", all = "logical" verbose = "logical") Cleans all spec-
    tra in MSnExp object. Displays a control bar if verbose set to TRUE (default). Returns a cleaned
    MSnExp instance.

signature(object = "Spectrum", all = "logical") Cleans the Spectrum object. Returns a
    cleaned Spectrum instance.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

removePeaks and trimMz for other spectra processing methods.

Examples

```
int <- c(1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0)
sp1 <- new("Spectrum2",
           intensity=int,
           mz=1:length(int))
sp2 <- clean(sp1) ## default is all=FALSE
intensity(sp1)
intensity(sp2)
intensity(clean(sp1, all = TRUE))

mz(sp1)
mz(sp2)
mz(clean(sp1, all = TRUE))

data(itraqdata)
itraqdata2 <- clean(itraqdata)
sum(peaksCount(itraqdata))
sum(peaksCount(itraqdata2))
processingData(itraqdata2)
```

combineFeatures	*Combines features in an* MSnSet *object*

Description

This function combines the features in an "MSnSet" instance applying a summarisation function
(see fun argument) to sets of features as defined by a factor (see groupBy argument). Note that the
feature names are automatically updated based on the groupBy parameter.

The coefficient of variations are automatically computed and collated to the featureData slot. See
cv and cv.norm arguments for details.

NB: All the functions available as fun take a na.rm argument. This argument is FALSE by default.
This will have as effect that NA get propagated at the higher level. It is generally advised to set
na.rm = TRUE. See the example below.

Usage

```
combineFeatures(object, groupBy, fun = c("mean", "median",
"weighted.mean", "sum", "medpolish", "iPQF"), redundancy.handler =
c("unique", "multiple"), cv = TRUE, cv.norm = "sum", verbose = TRUE,
...)
```

Arguments

object   An instance of class `"MSnSet"` whose features will be summerised.

groupBy   A `factor`, `character`, `numeric` or a `list` of the above defining how to sum-
                merise the features. The list must be of length nrow(object). Each element
                of the list is a vector describing the feature mapping. If the list can be named,
                its names must match fetureNames(object). See redundancy.handler for
                details about the latter.

fun   The summerising function. Currently, mean, median, weighted mean, sum and
                median polish and iPQF (see `iPQF` for details) are implemented, but user-defined
                functions can also be supplied.

redundancy.handler
                If groupBy is a `list`, one of `"unique"` (default) or `"multiple"` (ignored other-
                wise) defining how to handle peptides that can be associated to multiple higher-
                level features (proteins) upon combination. Using `"unique"` will only consider
                uniquely matching features (features matching multiple proteins will be dis-
                carded). `"multiple"` will allow matching to multiple proteins and each feature
                will be repeatedly tallied for each possible matching protein.

cv   A `logical` defining if feature coefficients of variation should be computed and
                stored as feature meta-data. Default is TRUE.

cv.norm   A `character` defining how to normalise the feature intensitites prior to CV cal-
                culation. Default is sum. Use none to keep intensities as is. See `featureCV` for
                more details.

verbose   A `logical` indicating whether verbose output is to be printed out.

...   Additional arguments for the `fun` function. In particular, for the iPQF method,
                additional arguments are described in the details section.

Value

A new `"MSnSet"` instance is returned with ncol (i.e. number of samples) is unchanged, but nrow
(i.e. the number od features) is now equals to the number of levels in groupBy. The feature metadata
(featureData slot) is updated accordingly and only the first occurrence of a feature in the original
feature meta-data is kept.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

iPQF: A new peptide-to-protein summarization method using peptide characteristics to improve
iTRAQ quantification Martina Fischer and Bernhard Y. Renard, *in prep*.

See Also

[featureCV](#)

Examples

```
data(msnset)
msnset <- msnset[11:15, ]
exprs(msnset)

## arbitrary grouping into two groups
grp <- as.factor(c(1, 1, 2, 2, 2))
msnset.comb <- combineFeatures(msnset, grp, "sum")
dim(msnset.comb)
exprs(msnset.comb)
fvarLabels(msnset.comb)

## grouping with a list
grpl <- list(c("A", "B"), "A", "A", "C", c("C", "B"))
## optional naming
names(grpl) <- featureNames(msnset)
exprs(combineFeatures(msnset, grpl, fun = "sum", redundancy.handler = "unique"))
exprs(combineFeatures(msnset, grpl, fun = "sum", redundancy.handler = "multiple"))

## missing data
exprs(msnset)[4, 4] <-
    exprs(msnset)[2, 2] <- NA
exprs(msnset)
## NAs propagate in the 115 and 117 channels
exprs(combineFeatures(msnset, grp, "sum"))
## NAs are removed before summing
exprs(combineFeatures(msnset, grp, "sum", na.rm = TRUE))

## using iPQF
data(msnset2)
res <- combineFeatures(msnset2,
                       groupBy = fData(msnset2)$accession,
                       redundancy.handler = "unique",
                       fun = "iPQF",
                       low.support.filter = FALSE,
                       ratio.calc = "sum",
                       method.combine = FALSE)
head(exprs(res))
```

commonFeatureNames   *Keep only common feature names*

Description

Subsets `MSnSet` instances to their common feature names.

Usage

```
commonFeatureNames(x, y)
```

Arguments

x   An instance of class [MSnSet](#) or a list of MSnSet instances of at least 2 MSnSet
             objects.

y   An instance of class [MSnSet](#). Ignored if x is a list of MSnSet instances.

Value

An `linkS4class{MSnSetList}` composed of the input `MSnSet` containing only common features
in the same order. The names of the output are either the names of the x and y input variables or the
names of x if a list is provided.

Author(s)

Laurent Gatto

Examples

```
library("pRolocdata")
data(tan2009r1)
data(tan2009r2)
cmn <- commonFeatureNames(tan2009r1, tan2009r2)
names(cmn)
## as a named list
names(commonFeatureNames(list(a = tan2009r1, b = tan2009r2)))
## without message
suppressMessages(cmn <- commonFeatureNames(tan2009r1, tan2009r2))
## more than 2 instance
data(tan2009r3)
cmn <- commonFeatureNames(list(tan2009r1, tan2009r2, tan2009r3))
length(cmn)
```

---

compareMSnSets   *Compare two MSnSets*

Description

Compares two [MSnSet](#) instances. The qual and processingData slots are generally omitted.

Usage

```
compareMSnSets(x, y, qual = FALSE, proc = FALSE)
```

Arguments

x	First MSnSet
y	Second MSnSet
qual	Should the qual slots be compared? Default is FALSE.
proc	Should the processingData slots be compared? Default is FALSE.

Value

A logical

Author(s)

Laurent Gatto

---

compareSpectra-methods

            Compare Spectra of an 'MSnExp' or 'Spectrum' instances

---

Description

This method compares spectra (Spectrum instances) pairwise or all spectra of an experiment (MSnExp instances). Currently the comparison is based on the number of common peaks fun = "common", the Pearson correlation fun = "cor", the dot product fun = "dotproduct" or a user-defined function.

For fun = "common" the tolerance (default 25e-6) can be set and the tolerance can be defined to be relative (default relative = TRUE) or absolute (relative = FALSE). To compare spectra with fun = "cor" and fun = "dotproduct", the spectra need to be binned. The binSize argument (in Dalton) controls the binning precision. Please see [bin](#) for details.

Instead of these three predefined functions for fun a user-defined comparison function can be supplied. This function takes two [Spectrum](#) objects as the first two arguments and ... as third argument. The function must return a single numeric value. See the example section.

Methods

signature(object1 = "MSnExp", object2 = "missing", fun = "character", ...) Compares
all spectra in an MSnExp object. The ... arguments are passed to the internal functions. Re-
turns a matrix of dimension length(object1) by length(object1).

signature(object1 = "Spectrum", object2 = "Spectrum", fun = "character", ...) Compares
two Spectrum objects. See the above explanation for fun and .... Returns a single numeric
value.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

References

Stein, S. E., & Scott, D. R. (1994). Optimization and testing of mass spectral library search algo-
rithms for compound identification. Journal of the American Society for Mass Spectrometry, 5(9),
859-866. doi: http://dx.doi.org/10.1016/1044-0305(94)87009-8

Lam, H., Deutsch, E. W., Eddes, J. S., Eng, J. K., King, N., Stein, S. E. and Aebersold, R. (2007)
Development and validation of a spectral library searching method for peptide identification from
MS/MS. Proteomics, 7: 655-667. doi: http://dx.doi.org/10.1002/pmic.200600625

See Also

bin, clean, pickPeaks, smooth, removePeaks and trimMz for other spectra processing methods.

Examples

```
s1 <- new("Spectrum2", mz=1:10, intensity=1:10)
s2 <- new("Spectrum2", mz=1:10, intensity=10:1)
compareSpectra(s1, s2)
compareSpectra(s1, s2, fun="cor", binSize=2)
compareSpectra(s1, s2, fun="dotproduct")

## define our own (useless) comparison function (it is just a basic example)
equalLength <- function(x, y, ...) {
  return(peaksCount(x)/(peaksCount(y)+.Machine$double.eps))
}
compareSpectra(s1, s2, fun=equalLength)
compareSpectra(s1, new("Spectrum2", mz=1:5, intensity=1:5), fun=equalLength)
compareSpectra(s1, new("Spectrum2"), fun=equalLength)

data(itraqdata)
compareSpectra(itraqdata[1:5], fun="cor")
```

exprsToRatios-methods   *Calculate all ratio pairs*

Description

Calculations all possible ratios for the assayData columns in an ["MSnSet"](). The function getRatios(x, log = FALSE) takes a matrix x as input and is used by exprsToRatios.

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.

signature(object = "matrix", log = "logical") As above, but for a matrix instance.

Examples

```
data(msnset)
pData(msnset)
head(exprs(msnset))
r <- exprsToRatios(msnset)
head(exprs(r))
pData(r)
```

extractPrecSpectra-methods
                    Extracts precursor-specific spectra from an 'MSnExp' object

Description

Extracts the MSMS spectra that originate from the precursor(s) having the same MZ value as defined in theprec argument.

A warning will be issued of one or several of the precursor MZ values in prec are absent in the experiment precursor MZ values (i.e in precursorMz(object)).

Methods

signature(object = "MSnExp", prec = "numeric") Returns an ["MSnExp"]() containing MSMS spectra whose precursor MZ values are in prec.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

```
file <- dir(system.file(package="MSnbase",dir="extdata"),
            full.name=TRUE,pattern="mzXML$")
aa <- readMSData(file,verbose=FALSE)
my.prec <- precursorMz(aa)[1]
my.prec
bb <- extractPrecSpectra(aa,my.prec)
precursorMz(bb)
processingData(bb)
```

FeatComp-class   *Class* "FeatComp"

Description

Comparing feature names of two comparable MSnSet instances.

Objects from the Class

Objects can be created with compfnames. The method compares the feature names of two objects of class "MSnSet". It prints a summary matrix of common and unique feature names and invisibly returns a list of FeatComp instances.

The function will compute the common and unique features for all feature names of the two input objects (featureNames(x) and feautreNames(y)) as well as distinct subsets as defined in the fcol1 and fcol2 feautre variables.

Slots

name: Object of class "character" defining the name of the compared features. By convention, "all" is used when all feature names are used; otherwise, the respective levels of the feature variables fcol1 and fcol2.

common: Object of class "character" with the common feature names.

unique1: Object of class "character" with the features unique to the first MSnSet (x in compfname).

unique2: Object of class "character" with the features unique to the seconn MSnSet (y in compfname).

all: Object of class "logical" defining if all features of only a subset were compared. One expects that name == "all" when all is TRUE.

Methods

Accessors names, common, unique1 and unique2 can be used to access the respective FeatComp slots.

compfnames signature(x = "MSnSet", y = "MSnSet", fcol1 = "character", fcol2 = "character", simplify = creates the FeatComp comparison object for instances x and y. The feature variables to be considered to details feature comparison can be defined by fcol1 (default is "markers" and fcol2 for x and y respectively). Setting either to NULL will only consider all feature names;

in such case, of simplify is TRUE (default), an FeatComp object is returned instead of a list of length 1. The verbose logical controls if a summary table needs to be printed (default is TRUE).

compfnames signature(x = "list", y = "missing", ...): when x is a list of MSnSet instances, compfnames is applied to all element pairs of x. Additional parameters fcol1, fcol2, simplify and verbose are passed to the pairwise comparison method.

show signature(object = "FeatComp"): prints a summary of the object.

Author(s)

Laurent Gatto <lg390@cam.ac.uk> and Thomas Naake

See Also

[averageMSnSet](#) to compute an average MSnSet.

Examples

```
library("pRolocdata")
data(tan2009r1)
data(tan2009r2)
x <- compfnames(tan2009r1, tan2009r2)
x[[1]]
x[2:3]
head(common(x[[1]]))

data(tan2009r3)
tanl <- list(tan2009r1, tan2009r2, tan2009r3)
xx <- compfnames(tanl, fcol1 = NULL)
length(xx)
tail(xx)

all.equal(xx[[15]],
          compfnames(tan2009r2, tan2009r3, fcol1 = NULL))
str(sapply(xx, common))
```

---

featureCV   *Calculates coeffivient of variation for features*

Description

This function calculates the column-wise coefficient of variation (CV), i.e. the ration between the standard deviation and the mean, for the features in an ["MSnSet"](#). The CVs are calculated for the groups of features defined by groupBy. For groups defined by single features, NA is returned.

Usage

```
featureCV(x, groupBy, na.rm = TRUE, norm = c("sum", "max", "none",
  "center.mean", "center.median", "quantiles", "quantiles.robust"))
```

Arguments

x	An instance of class `"MSnSet"`.
groupBy	An object of class `factor` defining how to summerise the features.
na.rm	A `logical` defining whether missing values should be removed.
norm	One of 'none' (default), 'sum', 'max', 'center.mean', 'center.median' 'quantiles' or 'quantiles.robust' defining if and how the data should be normalised prior to CV calculation. See `normalise` for more details.

Value

A `matrix` of dimensions `length(levels(groupBy))` by `ncol(x)` with the respecive CVs.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

`combineFeatures`

Examples

```
data(msnset)
msnset <- msnset[1:4]
gb <- factor(rep(1:2, each = 2))
featureCV(msnset, gb)
```

FeaturesOfInterest-class

Features of Interest

Description

The *Features of Interest* infrastructure allows to define a set of features of particular interest to be used/matched against existing data sets contained in `"MSnSet"`. A specific set of features is stored as an `FeaturesOfInterest` object and a collection of such non-redundant instances (for example for a specific organism, project, ...) can be collected in a `FoICollection`.

Objects from the Class

Objects can be created with the respective `FeaturesOfInterest` and `FoICollection` constructors.

`FeaturesOfInterest` instances can be generated in two different ways: the constructor takes either (1) a set of features names (a `character` vector) and a description (`character` of length 1 - any subsequent elements are silently ignored) or (2) feature names, a description and an instance of class `"MSnSet"`. In the latter case, we call such `FeaturesOfInterest` objects traceable, because we can identify the origin of the feature names and thus their validity. This is done by inspecting the MSnSet

instance and recording its dimensions, its name and a unique md5 hash tag (these are stores as part of the optional `objpar` slot). In such cases, the feature names passed to the `FeaturesOfInterest` constructor must also be present in the `MSnSet`; if one or more are not, an error will be thrown. If your features of interest to be recorded stem for an existing experiment and have all been observed, it is advised to pass the 3 arguments to the constructor to ensure that the feature names as valid. Otherwise, only the third argument should be omitted.

`FoICollection` instances can be constructed by creating an empty collection and serial additions of `FeaturesOfInterest` using `addFeaturesOfInterest` or by passing a list of `FeaturesOfInterest` instance.

Slots

`FeaturesOfInterest` class:

description: Object of class `"character"` describing the instance.

objpar: Optional object of class `"list"` providing details about the `MSnSet` instance originally used to create the instance. See details section.

fnames: Object of class `"character"` with the feature of interest names.

date: Object of class `"character"` with the date the instance was first generated.

.__classVersion__: Object of class `"Versions"` with the `FeaturesOfInterest` class version. Only relevant for development.

`FoICollection` class:

foic: Object of class `"list"` with the `FeaturesOfInterest`.

.__classVersion__: Object of class `"Versions"` with the `FoICollection` class version. Only relevant for development.

Extends

Class `"`[`Versioned`](#)`"`, directly.

Methods

`FeaturesOfInterest` class:

description `signature(object = "FeaturesOfInterest")`: returns the description of `object`.

foi `signature(object = "FeaturesOfInterest")`: returns the features of interests.

length `signature(x = "FeaturesOfInterest")`: returns the number of features of interest in `x`.

show `signature(object = "FeaturesOfInterest")`: displays `object`.

fnamesIn `signature(x = "FeaturesOfInterst", y = "MSnSet", count = "logical")`: if count is FALSE (default), return a logical indicating whether there is at least one feautre of interest present in `x`? Otherwise, returns the number of such features. Works also with matrices and data.frames.

`FoICollection` class:

description `signature(object = "FoICollection")`: returns the description of `object`.

foi signature(object = ″FoICollection″): returns a list of FeaturesOfInterest.

length signature(x = ″FoICollection″): returns the number of FeaturesOfInterest in the collection.

lengths signature(x = ″FoICollection″): returns the number of features of interest in each FeaturesOfInterest in the collection x.

addFeaturesOfInterest signature(x = ″FeaturesOfInterest″, y = ″FoICollection″): add the FeaturesOfInterest instance x to FoICollection y. If x is already present, a message is printed and y is returned unchanged.

rmFeaturesOfInterest signature(object = ″FoICollection″, i = ″numeric″): removes the ith FeatureOfInterest in the collection object.

show signature(object = ″FoICollection″): displays object.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

```
library("pRolocdata")
data(tan2009r1)

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

description(x)
foi(x)

y <- FeaturesOfInterest(description = "Non-traceable features of interest",
                        fnames = featureNames(tan2009r1)[111:113])
y

## an illegal FeaturesOfInterest
try(FeaturesOfInterest(description = "Won't work",
                       fnames = c("A", "Z", featureNames(tan2009r1)),
                       object = tan2009r1))


FeaturesOfInterest(description = "This work, but not traceable",
                   fnames = c("A", "Z", featureNames(tan2009r1)))


xx <- FoICollection()
xx

xx <- addFeaturesOfInterest(x, xx)
xx <- addFeaturesOfInterest(y, xx)
xx
```

```
description(xx)
foi(xx)

fnamesIn(x, tan2009r1)
fnamesIn(x, tan2009r1, count = TRUE)

rmFeaturesOfInterest(xx, 1)
```

---

fillUp   *Fills up a vector*

Description

This function replaces all the empty characters `""` and/or NAs with the value of the closest preceding
the preceding non-NA/`""` element. The function is used to populate dataframe or matrice columns
where only the cells of the first row in a set of partially identical rows are explicitly populated and
the following are empty.

Usage

```
fillUp(x)
```

Arguments

x   a vector.

Value

A vector as x with all empty characters `""` and NA values replaced by the preceding non-NA/`""` value.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

```
d <- data.frame(protein=c("Prot1","","","Prot2","",""),
                peptide=c("pep11","","pep12","pep21","pep22",""),
                score=c(1:2,NA,1:3))
d
e <- apply(d,2,fillUp)
e
data.frame(e)
fillUp(d[,1])
```

---

formatRt	*Format Retention Time*

Description

Converts seconds to/from 'min:sec' format

Usage

```
formatRt(rt)
```

Arguments

rt   retention in seconds (`numeric`) or "mm:sec" (`character`).

Details

This function is used to convert retention times. Conversion is seconds to/from the more human friendly format "mm:sec".

Value

A vector of same length as `rt`.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

```
formatRt(1524)
formatRt("25:24")
```

---

get.amino.acids	*Amino acids*

Description

Returns a `data.frame` of amino acid properties: `AA`, `ResidueMass`, `Abbrev3`, `ImmoniumIonMass`, `Name`, `Hydrophobicity`, `Hydrophilicity`, `SideChainMass`, `pK1`, `pK2` and `pI`.

Usage

```
get.amino.acids()
```

Value

A `data.frame`

Author(s)

Laurent Gatto

Examples

```
get.amino.acids()
```

`get.atomic.mass` *Atomic mass.*

Description

Returns a `double` of used atomic mass.

Usage

```
get.atomic.mass()
```

Value

A named double.

Author(s)

Sebastian Gibb

Examples

```
get.atomic.mass()
```

---

getVariableName *Return a variable name*

Description

Return the name of variable varname in call match_call.

Usage

```
getVariableName(match_call, varname)
```

Arguments

match_call   An object of class call, as returned by match.call.

varname   An character of length 1 which is looked up in match_call.

Value

A character with the name of the variable passed as parameter varname in parent close of match_call.

Author(s)

Laurent Gatto

Examples

```
a <- 1
f <- function(x, y)
 MSnbase:::getVariableName(match.call(), "x")
f(x = a)
f(y = a)
```

---

grepEcols *Returns the matching column names of indices.*

Description

Given a text spread sheet f and a pattern to be matched to its header (first line in the file), the function returns the matching columns names or indices of the corresponding data.frame.

Usage

```
grepEcols(f, pattern, ...)

getEcols(f, ...)
```

Arguments

f	A connection object or a `character` string to be read in with `readLines(f, n = 1)`.
pattern	A `character` string containing a regular expression to be matched to the file's header.
...	Additional parameters passed to `strsplit` to split the file header into individual column names.

Details

The function starts by reading the first line of the file (or connection) f with `readLines`, then splits it according to the optional `...` arguments (it is important to correctly specify `strsplit`'s split character vector here) and then matches `pattern` to the individual column names using `grep`.

Similarly, getEcols can be used to explore the column names and decide for the appropriate `pattern` value.

These functions are useful to check the parameters to be provided to `readMSnSet2`.

Value

Depending on `value`, the matching column names of indices. In case of getEcols, a `character` of column names.

Author(s)

Laurent Gatto

See Also

`readMSnSet2`

---

imageNA2 *NA heatmap visualisation for 2 groups*

Description

Produces a heatmap after reordring rows and columsn to highlight missing value patterns.

Usage

```
imageNA2(object, pcol, Rowv, Colv = TRUE, useGroupMean = FALSE,
  plot = TRUE, ...)
```

Arguments

`object`	An instance of class MSnSet
`pcol`	Either the name of a phenoData variable to be used to determine the group structure or a factor or any object that can be coerced as a factor of length equal to nrow(object). The resulting factor must have 2 levels. If missing (default) `image(object)` is called.
`Rowv`	Determines if and how the rows/features are reordered. If missing (default), rows are reordered according to `order((nNA1 + 1)^2/(nNA2 + 1))`, where NA1 and NA2 are the number of missing values in each group. Use a vector of numerics of feautre names to customise row order.
`Colv`	A `logical` that determines if columns/samples are reordered. Default is `TRUE`.
`useGroupMean`	Replace individual feature intensities by the group mean intensity. Default is FALSE.
`plot`	A `logical` specifying of an image should be produced. Default is `TRUE`.
`...`	Additional arguments passed to [image]().

Value

Used for its side effect of plotting. Invisibly returns Rovw and Colv.

Author(s)

Laurent Gatto, Samuel Wieczorek and Thomas Burger

Examples

```
library("pRolocdata")
library("pRoloc")
data(dunkley2006)
pcol <- ifelse(dunkley2006$fraction <= 5, "A", "B")
nax <- makeNaData(dunkley2006, pNA = 0.10)
exprs(nax)[sample(nrow(nax), 30), pcol == "A"] <- NA
exprs(nax)[sample(nrow(nax), 50), pcol == "B"] <- NA
MSnbase:::imageNA2(nax, pcol)
MSnbase:::imageNA2(nax, pcol, useGroupMean = TRUE)
MSnbase:::imageNA2(nax, pcol, Colv = FALSE, useGroupMean = FALSE)
MSnbase:::imageNA2(nax, pcol, Colv = FALSE, useGroupMean = TRUE)
```

---

impute-methods   *Quantitative proteomics data imputation*

Description

The `impute` method performs data imputation on an `MSnSet` instance using a variety of methods (see below). The imputation and the parameters are logged into the `processingData(object)` slot.

Users should proceed with care when imputing data and take precautions to assure that the imputation produce valid results, in particular with naive imputations such as replacing missing values with 0.

Details

There are two types of mechanisms resulting in missing values in LC/MSMS experiments.

- Missing values resulting from absence of detection of a feature, despite ions being present at detectable concentrations. For example in the case of ion suppression or as a result from the stochastic, data-dependent nature of the MS acquisition method. These missing value are expected to be randomly distributed in the data and are defined as missing at random (MAR) or missing completely at random (MCAR).

- Biologically relevant missing values resulting from the absence of the low abundance of ions (below the limit of detection of the instrument). These missing values are not expected to be randomly distributed in the data and are defined as missing not at random (MNAR).

MNAR features should ideally be imputed with a left-censor method, such as `QRILC` below. Conversely, it is recommended to use host deck methods such nearest neighbours, Bayesian missing value imputation or maximum likelihood methods when values are missing at random.

Currently, the following imputation methods are available:

MLE Maximum likelihood-based imputation method using the EM algorithm. Implemented in the norm::imp.norm function. See imp.norm for details and additional parameters. Note that here, `...` are passed to the em.norm function, rather to the actual imputation function `imp.norm`.

bcpa Bayesian missing value imputation are available, as implemented in the and pcaMethods::pca functions. See [pca](pca) for details and additional parameters.

knn Nearest neighbour averaging, as implemented in the impute::impute.knn function. See impute.knn for details and additional parameters.

QRILC A missing data imputation method that performs the imputation of left-censored missing data using random draws from a truncated distribution with parameters estimated using quantile regression. Implemented in the imputeLCMD::impute.QRILC function. See impute.QRILC for details and additional parameters.

MinDet Performs the imputation of left-censored missing data using a deterministic minimal value approach. Considering a expression data with *n* samples and *p* features, for each sample, the missing entries are replaced with a minimal value observed in that sample. The minimal value observed is estimated as being the q-th quantile (default q = 0.01) of the observed values in that sample. Implemented in the imputeLCMD::impute.MinDet function. See impute.MinDet for details and additional parameters.

MinProb Performs the imputation of left-censored missing data by random draws from a Gaussian distribution centred to a minimal value. Considering an expression data matrix with *n* samples

and *p* features, for each sample, the mean value of the Gaussian distribution is set to a minimal observed value in that sample. The minimal value observed is estimated as being the q-th quantile (default q = 0.01) of the observed values in that sample. The standard deviation is estimated as the median of the feature standard deviations. Note that when estimating the standard deviation of the Gaussian distribution, only the peptides/proteins which present more than 50% recorded values are considered. Implemented in the imputeLCMD::impute.MinProb function. See `impute.MinProb` for details and additional parameters.

min Replaces the missing values by the smallest non-missing value in the data.

zero Replaces the missing values by 0.

mixed A mixed imputation applying two methods (to be defined by the user as mar for values missing at random and mnar for values missing not at random, see example) on two M[C]AR/MNAR subsets of the data (as defined by the user by a randna logical, of length equal to nrow(object)).

nbavg Average neighbour imputation for fractions collected along a fractionation/separation gradient, such as sub-cellular fractions. The method assumes that the fraction are ordered along the gradient and is invalid otherwise.

Continuous sets NA value at the beginning and the end of the quantitation vectors are set to the lowest observed value in the data or to a user defined value passed as argument k. Them, when a missing value is flanked by two non-missing neighbouring values, it is imputed by the mean of its direct neighbours. A stretch of 2 or more missing values will not be imputed. See the example below.

The naset `MSnSet` is an real quantitative data where quantitative values have been replaced by NAs. See script/naset.R for details.

Methods

signature(object = "MSnSet", method, ...) This method performs data imputation on the object MSnSet instance using the method algorithm. ... is used to pass parameters to the imputation function. See the respective methods for details and additional parameters.

Author(s)

Laurent Gatto and Samuel Wieczorek

References

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein and Russ B. Altman, Missing value estimation methods for DNA microarrays Bioinformatics (2001) 17 (6): 520-525.

Oba et al., A Bayesian missing value estimation method for gene expression profile data, Bioinformatics (2003) 19 (16): 2088-2096.

Cosmin Lazar (2015). imputeLCMD: A collection of methods for left-censored missing data imputation. R package version 2.0. http://CRAN.R-project.org/package=imputeLCMD.

Examples

```
data(naset)
## table of missing values along the rows
table(fData(naset)$nNA)
## table of missing values along the columns
pData(naset)$nNA

## non-random missing values
notna <- which(!fData(naset)$randna)
length(notna)
notna

impute(naset, method = "min")

if (require("imputeLCMD")) {
    impute(naset, method = "QRILC")
    impute(naset, method = "MinDet")
}

if (require("norm"))
    impute(naset, method = "MLE")

impute(naset, "mixed",
       randna = fData(naset)$randna,
       mar = "knn", mnar = "QRILC")

## neighbour averaging

x <- naset[1:4, 1:6]
exprs(x)[1, 1] <- NA ## min value
exprs(x)[2, 3] <- NA ## average
exprs(x)[3, 1:2] <- NA ## min value and average
## 4th row: no imputation
exprs(x)

exprs(impute(x, "nbavg"))
```

---

iPQF   *iPQF: iTRAQ (and TMT) Protein Quantification based on Features*

Description

The iPQF spectra-to-protein summarisation method integrates peptide spectra characteristics and quantitative values for protein quantitation estimation. Spectra features, such as charge state, sequence length, identification score and others, contain valuable information concerning quantification accuracy. The iPQF algorithm assigns weights to spectra according to their overall feature reliability and computes a weighted mean to estimate protein quantities. See also combineFeatures for a more general overview of feature aggregation and examples.

Usage

```
iPQF(object, groupBy, low.support.filter = FALSE, ratio.calc = "sum",
  method.combine = FALSE, feature.weight = c(7, 6, 4, 3, 2, 1, 5)^2)
```

Arguments

object	An instance of class MSnSet containing absolute ion intensities.
groupBy	Vector defining spectra to protein matching. Generally, this is a feature variable such as fData(object)$accession.
low.support.filter	A logical specifying if proteins being supported by only 1-2 peptide spectra should be filtered out. Default is FALSE.
ratio.calc	Either "none" (don't calculate any ratios), "sum" (default), or a specific channel (one of sampleNames(object)) defining how to calculate relative peptides intensities.
method.combine	A logical defining whether to further use median polish to combine features.
feature.weight	Vector "numeric" giving weight to the different features. Default is the squared order of the features redundant -unique-distance metric, charge state, ion intensity, sequence length, identification score, modification state, and mass based on a robustness analysis.

Value

A matrix with estimated protein ratios.

Author(s)

Martina Fischer

References

iPQF: A new peptide-to-protein summarization method using peptide characteristics to improve iTRAQ quantification Martina Fischer and Bernhard Y. Renard, *in prep*.

Examples

```
data(msnset2)
head(exprs(msnset2))
prot <- combineFeatures(msnset2,
                        groupBy = fData(msnset2)$accession,
                        fun = "iPQF")
head(exprs(prot))
```

---

iTRAQ4 *iTRAQ 4-plex set*

Description

This instance of class *"*ReporterIons*"* corresponds to the iTRAQ 4-plex set, i.e the 114, 115, 116 and 117 isobaric tags. In the iTRAQ5 data set, an unfragmented tag, i.e reporter and attached isobaric tag, is also included at MZ 145. These objects are used to plot the reporter ions of interest in an MSMS spectra (see *"*Spectrum2*"*) as well as for quantification (see quantify).

Usage

```
iTRAQ4
iTRAQ5
iTRAQ8
iTRAQ9
```

References

Ross PL, Huang YN, Marchese JN, Williamson B, Parker K, Hattan S, Khainovski N, Pillai S, Dey S, Daniels S, Purkayastha S, Juhasz P, Martin S, Bartlet-Jones M, He F, Jacobson A, Pappin DJ. "Multiplexed protein quantitation in Saccharomyces cerevisiae using amine-reactive isobaric tagging reagents." *Mol Cell Proteomics*, 2004 Dec;3(12):1154-69. Epub 2004 Sep 22. PubMed PMID: 15385600.

See Also

TMT6.

Examples

```
iTRAQ4
iTRAQ4[1:2]

newReporter <- new("ReporterIons",
                   description="an example",
                   name="my reporter ions",
                   reporterNames=c("myrep1","myrep2"),
                   mz=c(121,122),
                   col=c("red","blue"),
                   width=0.05)
newReporter
```

---

itraqdata   *Example* MSnExp *data set*

Description

`itraqdata` is and example data sets is an iTRAQ 4-plex experiment that has been run on an Orbitrap Velos instrument. It includes identification data in the feature data slot obtain from the Mascot search engine. It is a subset of an spike-in experiment where proteins have spiked in an *Erwinia* background, as described in Karp et al. (2010), *Addressing accuracy and precision issues in iTRAQ quantitation*, Mol Cell Proteomics. 2010 Sep;9(9):1885-97. Epub 2010 Apr 10. (PMID 20382981). The spiked-in proteins in `itradata` are BSA and ENO and are present in relative abundances 1, 2.5, 5, 10 and 10, 5, 2.5, 1 in the 114, 115, 116 and 117 reporter tags. The `msnset` object is produced by running the `quantify` method on the `itraqdata` experimental data, as detailed in the [quantify](#) example. This example data set is used in the MSnbase-demo vignette, available with `vignette("MSnbase-demo", package="MSnbase")`.

The `msnset2` object is another example iTRAQ4 data that is used to demonstrate features of the package, in particular the `iPQF` feature aggregation method, described in [iPQF](#). It corresponds to 11 proteins with spectra measurements from the original data set described by Breitwieser et al. (2011) *General statistical modeling of data from protein relative expression isobaric tags*. J. Proteome Res., 10, 2758-2766.

Usage

```
itraqdata
```

Examples

```
data(itraqdata)
itraqdata

## created by
## msnset <- quantify(itraqdata, method = "trap", reporters = iTRAQ4)
data(msnset)
msnset

data(msnset2)
msnset2
```

---

listOf   *Tests equality of list elements class*

Description

Compares equality of all members of a list.

Usage

```
listOf(x, class, valid = TRUE)
```

Arguments

x	A codelist.
class	A character defining the expected class.
valid	A logical defining if all elements should be tested for validity. Default is TRUE.

Value

TRUE is all elements of x inherit from class.

Author(s)

Laurent Gatto

Examples

```
listOf(list(), "foo")
listOf(list("a", "b"), "character")
listOf(list("a", 1), "character")
```

---

makeMTD   *Creates the mzTab metadata section*

Description

mzTab is a light-weight, tab-delimited file format for proteomics data. It describes general metadata, protein, peptide and small molecule information (all of which are optinal), including quantitation and identification. The metadata section (MTD) can be generated from an MSnSet instance using makeMTD. The detailed description of all the parameters can be found in the mzTab specification document (see references).

Usage

```
makeMTD(x, unitId = NULL, title = NULL, mtdDescription = NULL,
  sampleProcessing = NULL, instrumentSource = NULL,
  instrumentAnalyzer = NULL, instrumentDetector = NULL, software = NULL,
  fdr = NULL, publication = NULL, contactName = NULL,
  contactAffiliation = NULL, contactEmail = NULL, mtdUri = NULL,
  mtdModifications = NULL, modProbabilityMethod = NULL,
  quantitationMethod = NULL, protQuantUnit = NULL, pepQuantUnit = NULL,
  msFileFormat = NULL, msFileLocation = NULL, msFileIdFormat = NULL,
  custom = NULL, species_ = NULL, tissue_ = NULL, cellType_ = NULL,
  disease_ = NULL, description_ = NULL, quantitationReagent_ = NULL,
  custom_ = NULL)
```

Arguments

x	An instance of class `MSnSet`.
unitId	A `character` of lenth 1 or NULL (default), in which case x's variable name will be used. This identifier references the item under study all sections.
title	A `character` of lenght 1 or NULL (default), in which case `exptitle(x)` is used if available.
mtdDescription	A `character` of length 1 describing the unit or NULL (default) to ignore.
sampleProcessing	
	A `list` of (possibly multiple) valid CVParam objects or NULL (default) to ignore.
instrumentSource	
	A list of valid `CVParam` instances or NULL (default), in which case `ionSource(x)` is used to generate a `CVParam`.
instrumentAnalyzer	
	A list of valid `CVParam` instances or NULL (default), in which case `analyzer(x)` is used to generate a `CVParam`.
instrumentDetector	
	A list of valid `CVParam` instances or NULL (default), in which case `detectorType(x)` is used to generate a `CVParam`.
software	A list of valid `CVParam` instances describing the ordered list of software used to process the data. NULL (default) to ignore.
fdr	A list of valid `CVParam` instances describing the unit's false discovery rate or NULL (default) to ignore.
publication	A `character` (of lenght > 0) or NULL (default), in which case `pubMedIds(x)` is used.
contactName	A `character` (of length > 0) or NULL (default), in which case `expinfo(x)["name"]` is used.
contactAffiliation	
	A `character` (of length > 0) or NULL (default), in which case `expinfo(x)["lab"]` is used.
contactEmail	A `character` (of length > 0) or NULL (default), in which case `expemail(x)` is used.
mtdUri	A `character` (of length > 0) describing the unit's uniform resource identifier (a PRIDE experiment or a PeptideAtlas build for example). NULL (default) to ignore.
mtdModifications	
	A list of (possibly multiplt) CVParam instances describing all (distinct) PTMs reported in the unit. NULL (default) to ignore.
modProbabilityMethod	
	A user defined CVParam reporting the modifiction (position) probabilities. NULL (default) to ignore.
quantitationMethod	
	A valid CVParam, a `ReporterIons` instance or NULL (default), in which case the isobaric tagging system is guessed from the number of columns in `exprs(x)` (4 or 8 for iTRAQ, 6 for TMT).

protQuantUnit	A valid `CVParam` or `NULL` (default) to use PRIDE:0000330 (Arbitrary quantificatio unit).
pepQuantUnit	A valid `CVParam` or `NULL` (default) to use PRIDE:0000330 (Arbitrary quantificatio unit).
msFileFormat	A list of valid `CVParam` instances to `NULL` (default), in which case, the extension of `fileNames(x)[1]` is used to define the appropriate `CVParam`. Recognised extensions are `mzData`, `mzXML`, `mzML` or `mgf`.
msFileLocation	A character (of length > 0) or `NULL` (default), in which case `fileNames(x)` is used.
msFileIdFormat	A list of `CVParam` instances describing the original identification format used in the external data file. `NULL` (default) to ignore.
custom	A list of user defined `CVParam` instances with additional parameters describing the unit. `NULL` (default) to ignore.
species_	A list of (possibly several) `CVParam` instances with the respective (sub-)unit species. `NULL` (default) to ignore.
tissue_	A list of (possibly several) `CVParam` instances describing the respective (sub-)unit tissue. `NULL` (default) to ignore.
cellType_	A list of (possibly several) `CVParam` instances describing the respective (sub-)unit cell type. `NULL` (default) to ignore.
disease_	A list of (possibly several) `CVParam` instances describing the respective (sub-)unit disease states. `NULL` (default) to ignore.
description_	A list of `characters` describing the (sub-)unit in human raedable free text. `NULL` (default) to ignore.
quantitationReagent_	
	A list of `CVParam` instances or `NULL` (default), in which case the reporter ions as defined by `quantitationMethod` as used.
custom_	A list of user defined `CVParam` instances with additional (sub-)unit properties. `NULL` (default) to ignore.

Value

A `character` defining the mzTab metadata section.

Author(s)

Laurent Gatto

References

mzTab - Reporting Proteomics Results (<http://code.google.com/p/mztab/>)

See Also

[makePEP](#) and [makePRT](#) to generate mzTab peptide and protein sections.

---

makePEP   *Creates the mzTab peptide section*

Description

mzTab is a light-weight, tab-delimited file format for proteomics data. It describes general metadata, protein, peptide and small molecule information (all of which are optinal), including quantitation and identification. The peptide section (PEH header and PEP tabular data) can be generated from an MSnSet instance using makePEP. The detailed description of all the parameters can be found in the mzTab specification document (see references).

Usage

```
makePEP(x, sequence = NA, pepAccession = NA, unitId = NULL, unique = NA,
  pepDatabase = NA, pepDatabaseVersion = NA, pepSearchEngine = NA,
  pepSearchEngineScore = NA, pepReliability = NA, pepModifications = NA,
  retentionTime = NA, charge = NA, massToCharge = NA, pepUri = NA,
  spectraRef = NA, pepAbundance = NULL, pepAbundanceStdev = NULL,
  pepAbundanceSterr = NULL, pepOpt_ = NULL)
```

Arguments

x	An instance of class MSnSet.
sequence	A character of length nrow(x) (will be recycled a whole number of times if of different length) with the peptide sequence. Default is NA.
pepAccession	A character of length nrow(x) (will be recycled a whole number of times if of different length) with the assigned protein accession. Default is NA.
unitId	A character of lenth 1 or NULL (default), in which case x's variable name will be used.
unique	A logical (converted to numeric to comply with format specification) of length(nrow(x) (will be recycled a whole number of times if of different length) specifying if peptide is proteotypic. Default is NA.
pepDatabase	A character of length nrow(x) (will be recycled a whole number of times if of different length) describing the protein database used for peptide identification. Default is NA.
pepDatabaseVersion	
	A character of length nrow(x) (will be recycled a whole number of times if of different length) with the database version. Default is NA.
pepSearchEngine	
	A list of length nrow(x) (of possibly multiple lists of) CVParam instances identifying the search engine used for peptide identification. Default is NA.
pepSearchEngineScore	
	A list of length nrow(x) (of possibly multiple lists of) CVParam instances specifying peptide identification scores. Default is NA.

pepReliability   A numeric of length nrow(x) (will be recycled a whole number of times if of
                 different length). Values should be 1 (high reliability), 2 (medium reliability) or
                 3 (poor reliability). Default is NA.

pepModifications

                 A character of length nrow(x) (will be recycled a whole number of times if
                 of different length) describing the modifications and their position (see mzTab
                 format specifications for details). Default is NA.

retentionTime   A numeric of length nrow(x) (will be recycled a whole number of times if of
                 different length). Note that currently, unique retention times are expected, but
                 could be extended to multiple times. Default is NA.

charge   A numeric of length nrow(x) (will be recycled a whole number of times if of
                 different length) indicating peptide charge state. Default is NA.

massToCharge   A numeric of length nrow(x) (will be recycled a whole number of times if of
                 different length) with the peptides precursor mass to charge ratio. Default is NA.

pepUri   A character of length nrow(x) (will be recycled a whole number of times
                 if of different length) with peptide uniform resource identifiers (link to PRIDE
                 database for instance). Default is NA.

spectraRef   A character in the format ms_file[1-n]:{SPEC_REF} (see mzTab specifica-
                 tions for details) of length nrow(x) (will be recycled a whole number of times
                 if of different length). Default is NA.

pepAbundance   A numeric of length nrow(x) or matrix with nrow(x) rows if multiple sub-
                 samples are reported (see metadata section), specifying the peptides abundance.
                 If NULL (default), ignored.

pepAbundanceStdev

                 A numeric of length nrow(x) or matrix with nrow(x) rows if multiple sub-
                 samples are reported (see metadata section), specifying the standard deviation
                 of peptides abundances. If NULL (default), ignored. If pepAbundance is not
                 NULL, then pepAbundanceStdev is NA if not specified.

pepAbundanceSterr

                 A numeric of length nrow(x) or matrix with nrow(x) rows if multiple sub-
                 samples are reported (see metadata section), specifying the standard error of
                 peptides abundances. If NULL (default), ignored. If pepAbundance is not NULL,
                 then pepAbundanceSterr is NA if not specified.

pepOpt_   An optional character of character matrix (possibly populated with text rep-
                 resenations of CVParam instances) for any custom peptide annotation. Default is
                 NULL to ignore.

Value

A data.frame defining the mzTab peptide section.

Author(s)

Laurent Gatto

See Also

[makeMTD](#) and [makePRT](#) to generate mzTab metadata and protein sections.

---

makePRT   *Creates the mzTab protein section*

Description

mzTab is a light-weight, tab-delimited file format for proteomics data. It describes general metadata, protein, peptide and small molecule information (all of which are optinal), including quantitation and identification. The proteine section (PRH header and PRT tabular data) can be generated from an [MSnSet](#) instance using makePRT. The detailed description of all the parameters can be found in the mzTab specification document (see references).

Usage

```
makePRT(x, protAccession = NA, unitId = NULL, protDescription = NA,
  taxId = NA, species = NA, protDatabase = NA, protDatabaseVersion = NA,
  protSearchEngine = NA, protSearchEngineScore = NA, protReliability = NA,
  numPep = NA, numPepDistinct = NA, numPepUnambiguous = NA,
  ambiguityMembers = NA, protModifications = NA, protUri = NA,
  goTerms = NA, protCoverage = NA, protAbundance = NULL,
  protAbundanceStdev = NULL, protAbundanceSterr = NULL, protOpt_ = NULL)
```

Arguments

x	An instance of class [MSnSet](#).
protAccession	A character of length nrow(x) (will be recycled a whole number of times if of different length) with the protein accession. Default is NA.
unitId	A character of lenth 1 or NULL (default), in which case x's variable name will be used.
protDescription	
	A character of length nrow(x) (will be recycled a whole number of times if of different length) with the protein name or description. Default is NA.
taxId	A numeric of length nrow(x) (will be recycled a whole number of times if of different length) referencing the species NCBI/NEWT taxonomy id. Default is NA.
species	A character of length nrow(x) (will be recycled a whole number of times if of different length) describing the species in human readable form. Default is NA.
protDatabase	A character of length nrow(x) (will be recycled a whole number of times if of different length) describing the protein database. Default is NA.
protDatabaseVersion	
	A character of length nrow(x) (will be recycled a whole number of times if of different length) describing the database version. Default is NA.

protSearchEngine

> A `list` of length `nrow(x)` (of possibly several) `CVParam` instances describing the search engine used for protein identification. Default is `NA`.

protSearchEngineScore

> A `list` of length `nrow(x)` (of possibly multiple lists of) `CVParam` instances specifying peptide identification scores. Default is `NA`.

protReliability

> A numeric of length `nrow(x)` (will be recycled a whole number of times if of different length). Values should be 1 (high reliability), 2 (medium reliability) or 3 (poor reliability). Default is `NA`.

numPep

> A numeric of length `nrow(x)` (will be recycled a whole number of times if of different length) indicating the number of peptides identifying the proteins. Default is `NA`.

numPepDistinct

> A numeric of length `nrow(x)` (will be recycled a whole number of times if of different length) indicating the number of distinct peptides (sequence and modifications) identifying the proteins. Default is `NA`.

numPepUnambiguous

> A numeric of length `nrow(x)` (will be recycled a whole number of times if of different length) indicating the number of unambiguous disctinct peptides identifying the proteins. Default is `NA`.

ambiguityMembers

> A `character` of comma-separated protein accessions. See the `mzTab` specification document for details. Defaut is `NA`.

protModifications

> A `character` of comma-delimited modifications/scores/positions describing the proteins. See the `mzTab`specification document for details. Defaut is `NA`.

protUri

> A `character` of length `nrow(x)` (will be recycled a whole number of times if of different length) with peptide uniform resource identifiers (link to PRIDE database for instance). Default is `NA`.

goTerms

> A `character` of length `nrow(x)` (will be recycled a whole number of times if of different length) with comma-delimited GO terms describing the proteins. Default is `NA`.

protCoverage

> A numeric of length `nrow(x)` (will be recycled a whole number of times if of different length) with the protein coverages ranging between 0 and 1. Default is `NA`.

protAbundance

> A numeric of length `nrow(x)` or `matrix` with `nrow(x)` rows if multiple subsamples are reported (see metadata section), specifying the protein abundance. If `NULL` (default), ignored.

protAbundanceStdev

> A numeric of length `nrow(x)` or `matrix` with `nrow(x)` rows if multiple subsamples are reported (see metadata section), specifying the standard deviation of protein abundances. If `NULL` (default), ignored. If `protAbundance` is not `NULL`, then `protAbundanceStdev` is `NA` if not specified.

protAbundanceSterr

> A numeric of length `nrow(x)` or `matrix` with `nrow(x)` rows if multiple subsamples are reported (see metadata section), specifying the standard error of

protein abundances. If NULL (default), ignored. If `protAbundance` is not NULL, then `protAbundanceSterr` is NA if not specified.

protOpt_   An optional `character` of `character` matrix (possibly populated with text represenations of `CVParam` instances) for any custom protein annotation. Default is NULL to ignore.

Value

A `data.frame` defining the `mzTab` protein section.

Author(s)

Laurent Gatto

References

mzTab - Reporting Proteomics Results (<http://code.google.com/p/mztab/>)

See Also

makeMTD and makePEP to generate `mzTab` metadat and peptide sections.

MIAPE-class   *The "MIAPE" Class for Storing Proteomics Experiment Information*

Description

The Minimum Information About a Proteomics Experiment. The current implementation is based on the MIAPE-MS 2.4 document.

Slots

title: Object of class `character` containing a single-sentence experiment title.

abstract: Object of class `character` containing an abstract describing the experiment.

url: Object of class `character` containing a URL for the experiment.

pubMedIds: Object of class `character` listing strings of PubMed identifiers of papers relevant to the dataset.

samples: Object of class `list` containing information about the samples.

preprocessing: Object of class `list` containing information about the pre-processing steps used on the raw data from this experiment.

other: Object of class `list` containing other information for which none of the above slots applies.

dateStamp: Object of class `character`, giving the date on which the work described was initiated; given in the standard 'YYYY-MM-DD' format (with hyphens).

name: Object of class `character` containing the name of the (stable) primary contact person for this data set; this could be the experimenter, lab head, line manager, . . .

lab: Object of class character containing the laboratory where the experiment was conducted.

contact: Object of class character containing contact information for lab and/or experimenter.

email: Object of class character containing tmail contact information for the primary contact person (see name above).

instrumentModel: Object of class character indicating the model of the mass spectrometer used to generate the data.

instrumentManufacturer: Object of class character indicating the manufacturing company of the mass spectrometer.

instrumentCustomisations: Object of class character describing any significant (i.e. affecting behaviour) deviations from the manufacturer's specification for the mass spectrometer.

softwareName: Object of class character with the instrument management and data analysis package(s) name(s).

softwareVersion: Object of class character with the instrument management and data analysis package(s) version(s).

switchingCriteria: Object of class character describing the list of conditions that cause the switch from survey or zoom mode (MS1) to or tandem mode (MSn where n > 1); e.g. 'parent ion" mass lists, neutral loss criteria and so on [applied for tandem MS only].

isolationWidth: Object of class numeric describing, for tandem instruments, the total width (i.e. not half for plus-or-minus) of the gate applied around a selected precursor ion m/z, provided for all levels or by MS level.

parameterFile: Object of class character giving the location and name under which the mass spectrometer's parameter settings file for the run is stored, if available. Ideally this should be a URI+filename, or most preferably an LSID, where feasible.

ionSource: Object of class character describing the ion source (ESI, MALDI, . . .).

ionSourceDetails: Object of class character describing the relevant details about the ion source. See MIAPE-MI docuement for more details.

analyser: Object of class character describing the analyzer type (Quadrupole, time-of-flight, ion trap, . . .).

analyserDetails: Object of class character describing the relevant details about the analyzer. See MIAPE-MI document for more details.

collisionGas: Object of class character describing the composition of the gas used to fragment ions in the collision cell.

collisionPressure: Object of class numeric providing the pressure (in bars) of the collision gas.

collisionEnergy: Object of class character specifying for the process of imparting a particular impetus to ions with a given m/z value, as they travel into the collision cell for fragmentation. This could be a global figure (e.g. for tandem TOF's), or a complex function; for example a gradient (stepped or continuous) of m/z values (for quads) or activation frequencies (for traps) with associated collision energies (given in eV). Note that collision energies are also provided for individual *"Spectrum2"* instances, and is the preferred way of accessing this data.

detectorType: Object of class character describing the type of detector used in the machine (microchannel plate, channeltron, . . .).

detectorSensitivity: Object of class character giving and appropriate measure of the sensitivity of the described detector (e.g. applied voltage).

Methods

The following methods as in `"MIAME"`:

`abstract(MIAPE)`**:** An accessor function for `abstract`.

`expinfo(MIAPE)`**:** An accessor function for `name`, `lab`, `contact`, `title`, and `url`.

`notes(MIAPE)`, `notes(MIAPE) <- value`**:** Accessor functions for `other`. `notes(MIAME) <- character` *appends* character to notes; use `notes(MIAPE) <- list` to replace the notes entirely.

`otherInfo(MIAPE)`**:** An accessor function for `other`.

`preproc(MIAPE)`**:** An accessor function for `preprocessing`.

`pubMedIds(MIAPE)`, `pubMedIds(MIAME) <- value`**:** Accessor function for `pubMedIds`.

`expemail(MIAPE)`**:** An accessor function for `email` slot.

`exptitle(MIAPE)`**:** An accessor function for `title` slot.

`analyzer(MIAPE)`**:** An accessor function for `analyser` slot. `analyser(MIAPE)` is also available.

`analyzerDetails(MIAPE)`**:** An accessor function for `analyserDetails` slot. `analyserDetails` is also available.

`detectorType(MIAPE)`**:** An accessor function for `detectorType` slot.

`ionSource(MIAPE)`**:** An accessor function for `ionSource` slot.

`ionSourceDetails(MIAPE)`**:** An accessor function for `ionSourceDetails` slot.

`instrumentModel(MIAPE)`**:** An accessor function for `instrumentModel` slot.

`instrumentManufacturer(MIAPE)`**:** An accessor function for `instrumentManufacturer` slot.

`instrumentCustomisations(MIAPE)`**:** An accessor function for `instrumentCustomisations` slot.

`as(,"MIAME")`**:** Coerce the object from `MIAPE` to `MIAME` class. Used when converting an `MSnSet` into an `ExpressionSet`.

MIAPE-specific methods, including MIAPE-MS meta-data:

`show(MIAPE)`**:** Displays the experiment data.

`msInfo(MIAPE)`**:** Displays 'MIAPE-MS' information.

Extends

Class `"MIAxE"`, directly. Class `"Versioned"`, by class "MIAxE", distance 2.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

About MIAPE: http://www.psidev.info/index.php?q=node/91, and references therein, especially 'Guidelines for reporting the use of mass spectrometry in proteomics', Nature Biotechnology 26, 860-861 (2008).

---

missing-data   *Documenting missing data visualisation*

Description

There is a need for adequate handling of missing value impuation in quantitative proteomics. Before developing a framework to handle missing data imputation optimally, we propose a set of visuali-sation tools. This document serves as an internal notebook for current progress and ideas that will eventually materialise in exported functionality in the MSnbase package.

Details

The explore the structure of missing values, we propose to

1. Explore missing values in the frame of the experimental design. The imageNA2 function offers such a simple visualisation. It is currently limited to 2-group designs/comparisons. In case of time course experiments or sub-cellular fractionation along a density gradient, we propose to split the time/gradient into 2 groups (early/late, top/bottom) as a first approximation.

2. Explore the proportion of missing values in each group.

3. Explore the total and group-wise feature intensity distributions.

The existing plotNA function illustrates the completeness/missingness of the data.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>, Samuel Wieczorek and Thomas Burger

See Also

plotNA, imageNA2.

Examples

```
## Other suggestions
library("pRolocdata")
library("pRoloc")
data(dunkley2006)
set.seed(1)
nax <- makeNaData(dunkley2006, pNA = 0.10)
pcol <- factor(ifelse(dunkley2006$fraction <= 5, "A", "B"))
sel1 <- pcol == "A"

## missing values in each sample
barplot(colSums(is.na(nax)), col = pcol)


## table of missing values in proteins
par(mfrow = c(3, 1))
barplot(table(rowSums(is.na(nax))), main = "All")
```

```
barplot(table(rowSums(is.na(nax)[sel1,])), main = "Group A")
barplot(table(rowSums(is.na(nax)[!sel1,])), main = "Group B")


fData(nax)$nNA1 <- rowSums(is.na(nax)[, sel1])
fData(nax)$nNA2 <- rowSums(is.na(nax)[, !sel1])
fData(nax)$nNA <- rowSums(is.na(nax))
o <- MSnbase:::imageNA2(nax, pcol)

plot((fData(nax)$nNA1 - fData(nax)$nNA2)[o], type = "l")
grid()

plot(sort(fData(nax)$nNA1 - fData(nax)$nNA2), type = "l")
grid()


o2 <- order(fData(nax)$nNA1 - fData(nax)$nNA2)
MSnbase:::imageNA2(nax, pcol, Rowv=o2)

layout(matrix(c(rep(1, 10), rep(2, 5)), nc = 3))
MSnbase:::imageNA2(nax, pcol, Rowv=o2)
plot((fData(nax)$nNA1 - fData(nax)$nNA)[o2], type = "l", col = "red",
     ylim = c(-9, 9), ylab = "")
lines((fData(nax)$nNA - fData(nax)$nNA2)[o2], col = "steelblue")
lines((fData(nax)$nNA1 - fData(nax)$nNA2)[o2], type = "l",
     lwd = 2)
```

MSmap-class   *Class* MSmap

Description

A class to store mass spectrometry data maps, i.e intensities collected along the M/Z and retention time space during a mass spectrometry acquisition.

Objects from the Class

Objects can be created with the MSmap constructor. The constructor has the following arguments:

object An object created by mzR::openMSfile.

scans A numeric indicating the scan indices to be extracted from object to create the MS map. If missing, all MS1 spectra will be used.

lowMz A numeric of length 1 defining the lower bound of the M/Z range of the MS map.

highMz A numeric of length 1 defining the upper bound of the M/Z range of the MS map.

resMz The resolution along the M/Z range.

hd An optional data.frame as produced by mzR::header(object). If missing, will be computer within the function.

zeroIsNA Set 0 intensities to NA. This can be used to clarify the 3 dimensional plot produce by plot3D.

Slots

 call: Object of class "call" - the call used to generate the instance.

 map: Object of class "matrix" containing the actual MS map.

 mz: Object of class "numeric" with the M/Z sampling bins.

 res: Object of class "numeric" storing the the M/Z resolution used to create the map.

 rt: Object of class "numeric" with the retention times of the map spectra.

 ms: Object of class "numeric" with the MS levels of the spectra.

 t: Object of class "logical" indicating if the instance has been transposed.

 filename: Object of class "character" specifying the filename of the original raw MS data.

Methods

 coerce signature(from = "MSmap", to = "data.frame"): convert the MSmap instance in a data.frame. Useful for plotting with lattice or ggplot2.

 fileName signature(object = "MSmap"): returns the raw data filename.

 msLevel signature(object = "MSmap"): returns the MS level of the map spectra.

 msMap signature(object = "MSmap"): returns the actual map matrix.

 mz signature(object = "MSmap", ...): returns the M/Z values of the map. Additional arguments are currently ignored.

 rtime signature(object = "MSmap", ...): returns retention time values of the map. Additional arguments are currently ignored.

 mzRes signature(object = "MSmap"): returns the resolution with which the sample along the M/Z range was done.

 dim signature(x = "MSmap"): returns the dimensions of the map. ncol and nrow return the number of columns and rows respectively.

 t signature(x = "MSmap"): transposes the map.

 show signature(object = "MSmap"): prints a summary of the map.

 plot signature(x = "MSmap", allTicks = "logical"): produces an image of the map using lattice::levelplot. By default, allTicks is TRUE and all M/Z and retention times ticks of drawn. If set to FALSE, only 10 ticks in each dimension are plotted.

 plot3D signature(object = "MSmap", rgl = "logical"): produces an three dimensional view of the map using lattice::cloude(..., type = "h"). If rgl is TRUE, the map is visualised on a rgl device and can be rotated with the mouse.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

```
## Not run:
    ## downloads the data
    library("rpx")
    px1 <- PXDataset("PXD000001")
    mzf <- pxget(px1, 6)

    ## reads the data
    ms <- openMSfile(mzf)
    hd <- header(ms)

    ## a set of spectra of interest: MS1 spectra eluted
    ## between 30 and 35 minutes retention time
    ms1 <- which(hd$msLevel == 1)
    rtsel <- hd$retentionTime[ms1] / 60 > 30 &
        hd$retentionTime[ms1] / 60 < 35

    ## the map
    M <- MSmap(ms, ms1[rtsel], 521, 523, .005, hd)

    plot(M, aspect = 1, allTicks = FALSE)
    plot3D(M)
    if (require("rgl") & interactive())
        plot3D(M, rgl = TRUE)

    ## With some MS2 spectra
    i <- ms1[which(rtsel)][1]
    j <- ms1[which(rtsel)][2]
    M2 <- MSmap(ms, i:j, 100, 1000, 1, hd)
    plot3D(M2)

## End(Not run)
```

MSnExp-class   *The 'MSnExp' Class for MS Data And Meta-Data*

Description

The MSnExp class encapsulates data and meta-data for mass spectrometry experiments, as described in the slots section. Several data files (currently in mzXML) can be loaded together with the function readMSData.

This class extends the virtual "pSet" class.

Objects from the Class

Objects can be created by calls of the form new("MSnExp",...). However, it is preferred to use the readMSData function that will read raw mass spectrometry data to generate a valid "MSnExp" instance.

Slots

assayData**:** Object of class "environment" containing the MS spectra (see "Spectrum1" and "Spectrum2"). Slot is inherited from "pSet".

phenoData**:** Object of class "AnnotatedDataFrame" containing experimenter-supplied variables describing sample (i.e the individual tags for an labelled MS experiment) See phenoData for more details. Slot is inherited from "pSet".

featureData**:** Object of class "AnnotatedDataFrame" containing variables describing features (spectra in our case), e.g. identificaiton data, peptide sequence, identification score,... (inherited from "eSet"). See featureData for more details. Slot is inherited from "pSet".

experimentData**:** Object of class "MIAPE", containing details of experimental methods. See experimentData for more details. Slot is inherited from "pSet".

protocolData**:** Object of class "AnnotatedDataFrame" containing equipment-generated variables (inherited from "eSet"). See protocolData for more details. Slot is inherited from "pSet".

processingData**:** Object of class "MSnProcess" that records all processing. Slot is inherited from "pSet".

.__classVersion__**:** Object of class "Versions" describing the versions of R, the Biobase package, "pSet" and MSnExp of the current instance. Slot is inherited from "pSet". Intended for developer use and debugging (inherited from "eSet").

Extends

Class "pSet", directly. Class "VersionedBiobase", by class "pSet", distance 2. Class "Versioned", by class "pSet", distance 3.

Methods

See the "pSet" class for documentation on accessors inherited from pSet, subsetting and general attribute accession.

bin signature(object = "MSnExp"): Bins spectra. See bin documentation for more details and examples.

clean signature(object = "MSnExp"): Removes unused 0 intensity data points. See clean documentation for more details and examples.

compareSpectra signature(object1 = "Spectrum",   object2 = "missing"): Compares spectra. See compareSpectra documentation for more details and examples.

extractPrecSpectra signature(object = "MSnExp", prec = "numeric"): extracts spectra with precursor MZ value equal to prec and returns an object of class 'MSnExp'. See extractPrecSpectra documentation for more details and examples.

pickPeaks signature(object = "MSnExp"): Performs the peak picking to generate centroided spectra. See pickPeaks documentation for more details and examples.

plot signature(x = "MSnExp", y = "missing"): Plots all the spectra of the MSnExp instance. See plot.MSnExp documentation for more details.

plot2d signature(object = "MSnExp", ...): Plots retention time against precursor MZ for MSnExp instances. See plot2d documentation for more details.

plotDensity signature(object = "MSnExp", ...): Plots the density of parameters of interest. instances. See [plotDensity](#) documentation for more details.

plotMzDelta signature(object = "MSnExp", ...): Plots a histogram of the m/z difference betwee all of the highest peaks of all MS2 spectra of an experiment. See [plotMzDelta](#) documentation for more details.

quantify signature(object = "MSnExp"): Performs quantification for all the MS2 spectra of the MSnExp instance. See [quantify](#) documentation for more details.

removePeaks signature(object = "MSnExp"): Removes peaks lower that a threshold t. See [removePeaks](#) documentation for more details and examples.

removeReporters signature(object = "MSnExp", ...): Removes reporter ion peaks from all MS2 spectra of an experiment. See [removeReporters](#) documentation for more details and examples.

smooth signature(x = "MSnExp"): Smooths spectra. See [smooth](#) documentation for more details and examples.

addIdentificationData signature(object = "MSnExp", ...): Adds identification data to an experiment. See [addIdentificationData](#) documentation for more details and examples.

removeNoId signature(object = "MSnExp", fcol = "pepseq", keep = NULL): Removes non-identified features. See [removeNoId](#) documentation for more details and examples.

removeMultipleAssignment signature(object = "MSnExp",fcol = "nprot"): Removes protein groups with more than one member. The latter is defined by extracting a feature variable (default is "nprot").

idSummary signature(object = "MSnExp", ...): Prints a summary that lists the percentage of identified features per file (called coverage).

show signature(object = "MSnExp"): Displays object content as text.

trimMz signature(object = "MSnExp"): Trims the MZ range of all the spectra of the MSnExp instance. See [trimMz](#) documentation for more details and examples.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

Information about the mzXML format as well converters from vendor specific formats to mzXML: http://tools.proteomecenter.org/wiki/index.php?title=Formats:mzXML.

See Also

["pSet"](#) and [readMSData](#) for loading mzXML, mzData or mzML files to generate an instance of MSnExp.

Examples

```
mzxmlfile <- dir(system.file("extdata",package="MSnbase"),
                 pattern="mzXML",full.names=TRUE)
msnexp <- readMSData(mzxmlfile)
msnexp
```

MSnProcess-class   *The "MSnProcess" Class*

Description

MSnProcess is a container for MSnExp and MSnSet processing information. It records data files, processing steps, thresholds, analysis methods and times that have been applied to MSnExp or MSnSet instances.

Slots

files: Object of class "character" storing the raw data files used in experiment described by the "MSnProcess" instance.

processing: Object of class "character" storing all the processing steps and times.

merged: Object of class "logical" indicating whether spectra have been merged.

cleaned: Object of class "logical" indicating whether spectra have been cleaned. See [clean](#) for more details and examples.

removedPeaks: Object of class "character" describing whether peaks have been removed and which threshold was used. See [removePeaks](#) for more details and examples.

smoothed: Object of class "logical" indicating whether spectra have been smoothed.

trimmed: Object of class "numeric" documenting if/how the data has been trimmed.

normalised: Object of class "logical" describing whether and how data have been normalised.

MSnbaseVersion: Object of class "character" indicating the version of MSnbase.

.__classVersion__: Object of class "Versions" indicating the version of the MSnProcess instance. Intended for developer use and debugging.

Extends

Class "[Versioned](#)", directly.

Methods

fileNames signature(object = "MSnProcess"): Returns the file names used in experiment described by the "MSnProcess" instance.

show signature(object = "MSnProcess"): Displays object content as text.

combine signature(x = "MSnProcess", y = "MSnProcess"): Combines multiple MSnProcess instances.

Note

This class is likely to be updated using an AnnotatedDataFrame.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

See the `"MSnExp"` and `"MSnSet"` classes that actually use MSnProcess as a slot.

Examples

```
showClass("MSnProcess")
```

MSnSet-class   *The "MSnSet" Class for MS Proteomics Expression Data and Meta-*
                                Data

Description

The MSnSet holds quantified expression data for MS proteomics data and the experimental meta-data. The MSnSet class is derived from the `"eSet"` class and mimics the `"ExpressionSet"` class classically used for microarray data.

Objects from the Class

The constructor MSnSet(exprs, fData, pData) can be used to create MSnSet instances. Argument exprs is a matrix and fData and pData must be of clas data.frame or `"AnnotatedDataFrame"` and all must meet the dimensions and name validity constrains.

Objects can also be created by calls of the form new("MSnSet", exprs, ...). See also `"ExpressionSet"` for helpful information. Expression data produced from other softwares can thus make use of this standardized data container to benefit R and Bioconductor packages. Importer functions will be developed to stream-line the generation of `"MSnSet"` instances from third-party software.

A coercion method is also available to transform an IBSpectra object (names x) from the isobar package into an MSnSet: as(x, "MSnSet").

In the frame of the MSnbase package, MSnSet instances can be generated from `"MSnExp"` experiments using the `quantify` method).

Slots

qual: Object of class `"data.frame"` that records peaks data for each of the reporter ions to be used as quality metrics.

processingData: Object of class `"MSnProcess"` that records all processing.

assayData: Object of class `"assayData"` containing a matrix with equal with column number equal to nrow(phenoData). assayData must contain a matrix exprs with rows represening features (e.g., reporters ions) and columns representing samples. See the `"AssayData"` class, `exprs` and `assayData` accessor for more details. This slot in indirectly inherited from `"eSet"`.

phenoData: Object of class `"AnnotatedDataFrame"` containing experimenter-supplied variables describing sample (i.e the individual tags for an labelled MS experiment) (indireclty inherited from `"eSet"`). See `phenoData` and the `"eSet"` class for more details.

featureData: Object of class "AnnotatedDataFrame" containing variables describing features (spectra in our case), e.g. identificaiton data, peptide sequence, identification score,... (inherited indirectly from "eSet"). See featureData and the "eSet" class for more details.

experimentData: Object of class "MIAPE", containing details of experimental methods (inherited from "eSet"). See experimentData and the "eSet" class for more details.

annotation: not used here.

protocolData: Object of class "AnnotatedDataFrame" containing equipment-generated variables (inherited indirectly from "eSet"). See protocolData and the "eSet" class for more details.

.__classVersion__: Object of class "Versions" describing the versions of R, the Biobase package, "eSet", "pSet" and MSnSet of the current instance. Intended for developer use and debugging (inherited indirectly from "eSet").

Extends

Class "eSet", directly. Class "VersionedBiobase", by class "eSet", distance 2. Class "Versioned", by class "eSet", distance 3.

Methods

MSnSet specific methods or over-riding it's super-class are described below. See also more "eSet" for inherited methods.

dim signature(x = "MSnSet"): Returns the dimensions of object's assay data, i.e the number of samples and the number of features.

fileNames signature(object = "MSnSet"): Access file names in the processingData slot.

msInfo signature(object = "MSnSet"): Prints the MIAPE-MS meta-data stored in the experimentData slot.

processingData signature(object = "MSnSet"): Access the processingData slot.

show signature(object = "MSnSet"): Displays object content as text.

qual signature(object = "MSnSet"): Access the reporter ion peaks description.

purityCorrect signature(object = "MSnSet", impurities = "matrix"): performs reporter ions purity correction. See purityCorrect documentation for more details.

normalise signature(object = "MSnSet"): Performs MSnSet normalisation. See normalise for more details.

t signature(x = "MSnSet"): Returns a transposed MSnSet object where features are now aligned along columns and samples along rows and the phenoData and featureData slots have been swapped. The protocolData slot is always dropped.

as("ExpressionSet") signature(x = "MSnSet"): Coerce object from MSnSet to ExpressionSet-class. The experimentData slot is converted to a MIAME instance. It is also possible to coerce an ExpressionSet to and MSnSet, in which case the experimentData slot is newly initialised.

as("data.frame") signature(x = "MSnSet"): Coerce object from MSnSet to data.frame. The MSnSet is transposed and the PhenoData slot is appended. See also ms2df below.

write.exprs signature(x = "MSnSet")Writes expression values to a tab-separated file (default is tmp.txt). The fDataCols parameter can be used to specify which featureData columns (as column names, column number or logical) to append on the right of the expression matrix. The following arguments are the same as write.table.

combine signature(x = "MSnSet", y = "MSnSet", ...) Combines 2 or more MSnSet instances according to their feature names. Note that the qual slot and the processing information are silently dropped.

topN signature(object = "MSnSet", groupBy, n = 3, fun, ...) Selects the n most intense features (typically peptides or spectra) out of all available for each set defined by groupBy (typically proteins) and creates a new instance of class MSnSet. If less than n features are available, all are selected. The ncol(object) features are summarised using fun (default is sum) prior to be ordered in decreasing order. Additional parameters can be passed to fun through ..., for instance to control the behaviour of topN in case of NA values. Note that the qual slot and the processing information are silently dropped. (Works also with matrix instances.)

> See also the [nQuants](#) function to retrieve the actual number of retained peptides out of n.

> A complete use case using topN and nQuants is detailed in the synapter package vignette.

filterNA signature(object = "MSnSet", pNA = "numeric", pattern = "character", droplevels = "logical") This method subsets object by removing features that have (strictly) more than pNA percent of NA values. Default pNA is 0, which removes any feature that exhibits missing data. The method can also be used with a character pattern composed of 0 or 1 characters only. A 0 represent a column/sample that is allowed a missing values, while columns/samples with and 1 must not have NAs.

> This method also accepts matrix instances. droplevels defines whether unused levels in the feature meta-data ought to be lost. Default is TRUE. See the droplevels method below.

> See also the [is.na.MSnSet](#) and [plotNA](#) methods for missing data exploration.

filterZero signature(object = "MSnSet", pNA = "numeric", pattern = "character", droplevels = "logical") As filterNA, but for zeros.

log signature(object = "MSnSet", base = "numeric") Log transforms exprs(object) using base::log. base (defaults is e='exp(1)') must be a positive or complex number, the base with respect to which logarithms are computed.

droplevels signature(x = "MSnSet", ...)Drops the unused factor levels in the featureData slot. See [droplevels](#) for details.

exprsToRatios signature(object = "MSnSet", log = "logical")calculates all possible ratios between object's columns/samples. See [exprsToRatios](#) for more details.

impute signature(object = "MSnSet", ...) Performs data imputation on the MSnSet object. See [impute](#) for more details.

Additional accessors for the experimental metadata (experimentData slot) are defined. See ["MIAPE"](#) for details.

Plotting

meanSdPlot signature(object = "MSnSet") Plots row standard deviations versus row means. See [meanSdPlot](#) (vsn package) for more details.

image signature(x = "MSnSet", facetBy = "character",sOrderBy = "character", legend = "character", low "numeric")Produces an heatmap of expression values in the x object. Simple horizontal facetting is enabled by passing a single character as facetBy. Arbitrary facetting can be performed manually by saving the return value of the method (see example below). Re-ordering of the samples is possible by providing the name of a phenotypic variable to sOrderBy. The title of the legend can be set with legend and the colours with the low and high arguments.

> If any negative value is detected in the data, the values are considered as log fold-changes and a divergent colour scale is used. Otherwise, a gradient from low to high is used. To scale the quantitative data in x prior to plotting, please see the scale method.
>
> When there are more than nmax (default is 50) features/rows, these are not printed. This behaviour can be controlled by setting fnames to TRUE (always print) or FALSE (never print). See examples below.
>
> The code is based on Vlad Petyuk's vp.misc::image_msnset. The previous version of this method is still available through the image2 function.

plotNA signature(object = "MSnSet", pNA = "numeric") Plots missing data for an MSnSet instance. pNA is a numeric of length 1 that specifies the percentage of accepted missing data values per features. This value will be highlighted with a point on the figure, illustrating the overall percentage of NA values in the full data set and the number of proteins retained. Default is 1/2. See also plotNA.

MAplot signature(object = "MSnSet", log.it = "logical",   base = "numeric", ...) Produces MA plots (Ratio as a function of average intensity) for the samples in object. If ncol(object) == 2, then one MA plot is produced using the ma.plot function from the affy package. If object has more than 2 columns, then mva.pairs. log.it specifies is the data should be log-transformed (default is TRUE) using base. Further ... arguments will be passed to the respective functions.

addIdentificationData signature(object = "MSnSet", ...): Adds identification data to a MSnSet instance. See addIdentificationData documentation for more details and examples.

removeNoId signature(object = "MSnSet", fcol = "pepseq", keep = NULL): Removes non-identified features. See removeNoId documentation for more details and examples.

removeMultipleAssignment signature(object = "MSnSet",fcol = "nprot"): Removes protein groups with more than one member. The latter is defined by extracting a feature variable (default is "nprot").

idSummary signature(object = "MSnSet", ...): Prints a summary that lists the percentage of identified features per file (called coverage).

Functions

updateFvarLabels signature(object, label, sep) This function updates object's featureData variable labels by appending label. By default, label is the variable name and the separator sep is ..

updateSampleNames signature(object, label, sep) This function updates object's sample names by appending label. By default, label is the variable name and the separator sep is ..

updateFeatureNames signature(object, label, sep) This function updates object's feature names by appending label. By default, label is the variable name and the separator sep is ..

ms2df signature(x, fcols)Coerces the MSnSet instance to a data.frame. The direction of the data is retained and the feature variable labels that match fcol are appended to the expression values. See also as(x, "data.frame") above.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

"eSet", "ExpressionSet" and quantify. MSnSet quantitation values can be exported to a file with write.exprs.

Examples

```
data(msnset)
msnset <- msnset[10:15]

exprs(msnset)[1, c(1, 4)] <- NA
exprs(msnset)[2, c(1, 2)] <- NA
is.na(msnset)
featureNames(filterNA(msnset, pNA = 1/4))
featureNames(filterNA(msnset, pattern = "0110"))

M <- matrix(rnorm(12), 4)
pd <- data.frame(otherpdata = letters[1:3])
fd <- data.frame(otherfdata = letters[1:4])
x0 <- MSnSet(M, fd, pd)
sampleNames(x0)

M <- matrix(rnorm(12), 4)
colnames(M) <- LETTERS[1:3]
rownames(M) <- paste0("id", LETTERS[1:4])
pd <- data.frame(otherpdata = letters[1:3])
rownames(pd) <- colnames(M)
fd <- data.frame(otherfdata = letters[1:4])
rownames(fd) <- rownames(M)
x <- MSnSet(M, fd, pd)
sampleNames(x)


## Visualisation

library(pRolocdata)
data(dunkley2006)
image(dunkley2006)
## Changing colours
image(dunkley2006, high = "darkgreen")
image(dunkley2006, high = "darkgreen", low = "yellow")
## Forcing feature names
image(dunkley2006, fnames = TRUE)
## Facetting
image(dunkley2006, facetBy = "replicate")
p <- image(dunkley2006)
library("ggplot2") ## for facet_grid
p + facet_grid(replicate ~ membrane.prep, scales = 'free', space = 'free')
p + facet_grid(markers ~ replicate)
## Fold-changes
dd <- dunkley2006
exprs(dd) <- exprs(dd) - 0.25
image(dd)
```

```
image(dd, low = "green", high = "red")
## Feature names are displayed by default for smaller data
dunkley2006 <- dunkley2006[1:25, ]
image(dunkley2006)
image(dunkley2006, legend = "hello")
```

MSnSetList-class   *Storing multiple related MSnSets*

Description

A class for storing lists of MSnSet instances.

Details

There are two ways to store different sets of measurements pertaining an experimental unit, such as replicated measures of different conditions that were recorded over more than one MS acquisition. Without focusing on any proteomics technology in particular, these multiple assays can be recorded as

- A single combined MSnSet (see the section *Combining MSnSet instances* in the *MSnbase-demo* section). In such cases, the different experimental (phenotypical) conditions are recorded as an AnnotatedDataFrame in the phenoData slots.

  Quantitative data for features that were missing in an assay are generally encode as missing with NA values. Alternatively, only features observed in all assays could be selected. See the commonFeatureNames functions to select only common features among two or more MSnSet instance.

- Each set of measurements is stored in an MSnSet which are combined into one MSnSetList. Each MSnSet elements can have identical or different samples and features. Unless compiled directly manually by the user, one would expect at least one of these dimensions (features/rows or samples/columns) are conserved (i.e. all feature or samples names are identical). See split/unsplit below.

Objects from the Class

Objects can be created and manipluated with:

MSnSetList(x) The class constructor that takes a list of valid MSnSet instances as input x.

split(x, f) An MSnSetList can be created from an MSnSet instance. x is a single MSnSet and f is a factor or a character of length 1. In the latter case, f will be matched to the feature- and phenodata variable names (in that order). If a match is found, the respective variable is extracted, converted to a factor if it is not one already, and used to split x along the features/rows (f was a feature variable name) or samples/columns (f was a phenotypic variable name). If f is passed as a factor, its length will be matched to nrow(x) or ncol(x) (in that order) to determine if x will be split along the features (rows) or sample (columns). Hence, the length of f must match exactly to either dimension.

unsplit(value, f) The unsplit method reverses the effect of splitting the value MSnSet along the groups f.

as(x, "MSnSetList") Where x is an instance of class [MzTab](). See the class documentation for details.

Slots

x: Object of class list containing valid MSnSet instances. Can be extracted with the msnsets() accessor.

log: Object of class list containing an object creation log, containing among other elements the call that generated the object. Can be accessed with objlog().

.__classVersion__: The version of the instance. For development purposes only.

Methods

"[[" Extracts a single MSnSet at position.

"[" Extracts one of more MSnSets as MSnSetList.

length Returns the number of MSnSets.

names Returns the names of MSnSets, if 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 MSnSet, then the return value is an MSnSetList, otherwise a list.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

The [commonFeatureNames]() function to select common features among MSnSet instances.

Examples

```
library("pRolocdata")
data(tan2009r1)
data(tan2009r2)

## The MSnSetList class
##  an unnamed example
msnl <- MSnSetList(list(tan2009r1, tan2009r2))
##  a named example
msnl <- MSnSetList(list(A = tan2009r1, B = tan2009r2))
names(msnl)
msnsets(msnl)
length(msnl)
objlog(msnl)
msnl[[1]] ## an MSnSet
msnl[1]   ## an MSnSetList of length 1
```

```
## Iterating over the elements
lapply(msnl, dim) ## a list
lapply(msnl, normalise) ## an MSnSetList

## Splitting and unsplitting
##  splitting along the columns/samples
data(dunkley2006)
head(pData(dunkley2006))
(splt <- split(dunkley2006, "replicate"))
lapply(splt, dim) ## the number of rows and columns of the split elements
unsplt <- unsplit(splt, dunkley2006$replicate)
stopifnot(compareMSnSets(dunkley2006, unsplt))

##  splitting along the rows/features
head(fData(dunkley2006))
(splt <- split(dunkley2006, "markers"))
unsplt <- unsplit(splt, factor(fData(dunkley2006)$markers))
simplify2array(lapply(splt, dim))
stopifnot(compareMSnSets(dunkley2006, unsplt))
```

MzTab-class   *Parse* MzTab *files*

Description

The MzTab class stores the output of a basic parsing of a mzTab file. It contain the metadata (a list),
comments (a character vector), and the at least of of the following data types: proteins, peptides,
PSMs and small molecules (as data.frames).

At this stage, the metadata and data are only minimally parsed. More specific data extraction and
preparation are delegated to more specialised functions, such as the as(., to = "MSnSetList")
and readMzTabData for proteomics data.

Note that no attempts are made to verify the validitiy of the mzTab file.

Objects from the Class

Objects can be created by calls the the constructor MzTab that takes a single mzTab file as input.

The objects can subsequently be coerced to [MSnSetList](#) instances with as(object, "MSnSetList").
The resulting MSnSetList contains possibly empty MSnSet instances for proteins, peptide and
PSMs, respectively named "Proteins", "Peptides" and "PSMs".

The assaydata slots of the two former are populated with the protein_abundance_assay[1-n] and
peptide_abundance_assay[1-n] columns in the mzTab file. No abundance values are defined for
the latter. The respective feature names correspond to protein accessions, peptide sequences and
PSM identifiers, possibly made unique as by appending sequence numbers to duplicates.

Slots

Metadata: Object of class "list" storing the metadata section.

Filename: Object of class "character" storing the orginal file name.

Proteins: Object of class "data.frame" storing the protein data.

Peptides: Object of class "data.frame" storing the peptide data.

PSMs: Object of class "data.frame" storing the PSM data.

SmallMolecules: Object of class "data.frame" storing the small molecules data.

Comments: Object of class "character" storing the comments that were present in the file.

Accessors

metadata signature(x = "MzTab"): returns the meta data list.

mzTabMode signature(x = "MzTab"): returns the mode (complete or summary) of the mzTab data. A shortcut for metadata(x)$`mzTab-mode`.

mzTabType signature(x = "MzTab"): returns the type (quantification or identification) of the mzTab data. A shortcut for metadata(x)$`mzTab-type`.

fileName signature(object = "MzTab"): returns the file name of the original mzTab file.

peptides signature(object = "MzTab"): returns the peptide data.frame.

proteins signature(object = "MzTab"): returns the proteins data.frame.

psms signature(object = "MzTab"): returns the PSMs data.frame.

smallMolecules signature(object = "MzTab"): returns the small molecules data.frame.

comments signature(object = "MzTab"): returns the comments.

Author(s)

Laurent Gatto, with contributions from Richard Cotton (see https://github.com/lgatto/MSnbase/issues/41).

References

The mzTab format is a light-weight, tab-delimited file format for proteomics data. See https://code.google.com/p/mztab/ for details and specifications.

Griss J, Jones AR, Sachsenberg T, Walzer M, Gatto L, Hartler J, Thallinger GG, Salek RM, Steinbeck C, Neuhauser N, Cox J, Neumann S, Fan J, Reisinger F, Xu QW, Del Toro N, Perez-Riverol Y, Ghali F, Bandeira N, Xenarios I, Kohlbacher O, Vizcaino JA, Hermjakob H. The mzTab data exchange format: communicating mass-spectrometry-based proteomics and metabolomics experimental results to a wider audience. Mol Cell Proteomics. 2014 Oct;13(10):2765-75. doi: 10.1074/mcp.O113.036681. Epub 2014 Jun 30. PubMed PMID: 24980485; PubMed Central PMCID: PMC4189001.

Examples

```
## Test files from the mzTab developement repository
fls <- c("Cytidine.mzTab", "MTBLS2.mztab",
         "PRIDE_Exp_Complete_Ac_1643.xml-mztab.txt",
         "PRIDE_Exp_Complete_Ac_16649.xml-mztab.txt",
         "SILAC_CQI.mzTab", "SILAC_SQ.mzTab",
         "iTRAQ_CQI.mzTab", "iTRAQ_SQI.mzTab",
         "labelfree_CQI.mzTab", "labelfree_SQI.mzTab",
         "lipidomics-HFD-LD-study-PL-DG-SM.mzTab",
         "lipidomics-HFD-LD-study-TG.mzTab")
baseUrl <- "https://mztab.googlecode.com/svn/examples/"

## a list of mzTab objects
mzt <- sapply(file.path(baseUrl, fls), MzTab)
stopifnot(length(mzt) == length(fls))
mzt[[4]]

dim(proteins(mzt[[4]]))
dim(psms(mzt[[4]]))

prots4 <- proteins(mzt[[4]])
class(prots4)
prots4[1:5, 1:4]
```

NAnnotatedDataFrame-class

Class Containing Measured Variables and Their Meta-Data Description for Multiplexed Experiments.

Description

An NAnnotatedDataFrame is an "AnnotatedDataFrame", as defined in the 'Biobase' package that includes additional labels for multiplexing annotation.

Objects from the Class

See "AnnotatedDataFrame" for object creation with new. Multiplexing data is defined by setting the multiplex and multiLables paramters.

Slots

multiplex: Object of class "numeric" indicating the number of multiplexed samples described.

multiLabels: Object of class "character" describing the multiplexing.

varMetadata: Object of class "data.frame" with number of rows equal number of columns in data, and at least one column, named labelDescription, containing a textual description of each variable. Inherited from "AnnotatedDataFrame".

data: Object of class "data.frame" containing samples (rows) and measured variables (columns). Inherited from "AnnotatedDataFrame".

dimLabels: Object of class ″character″ of length 2 that provides labels for the rows and columns in the show method. Inherited from ″AnnotatedDataFrame″.

.__classVersion__: Object of class ″Versions″ describing the instance version. Intended for developer use. Inherited from ″AnnotatedDataFrame″.

Extends

Class ″AnnotatedDataFrame″, directly. Class ″Versioned″, by class "AnnotatedDataFrame", distance 2.

Methods

dim signature(object = ″NAnnotatedDataFrame″): Returns the number of samples, variables and multiplex cardinality in the object.

multiplex signature(object = ″NAnnotatedDataFrame″): Returns the number of multipexed samples described by the object.

multiLabels signature(object = ″NAnnotatedDataFrame″): Returns the multiplex labels.

show signature(object = ″NAnnotatedDataFrame″): Textual description of the object.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

″AnnotatedDataFrame″.

Examples

```
df <- data.frame(x=1:3,
                 y=LETTERS[1:3],
                 row.names=paste("Sample",1:3,sep=""))
metaData <-
  data.frame(labelDescription=c(
               "Numbers",
               "Factor levels"))
mplx <- c("M1","M2")
new("NAnnotatedDataFrame",
    data=df,
    varMetadata=metaData,
    multiplex=length(mplx),
    multiLabels=mplx)
```

normalise-methods   *Normalisation of* MSnExp, MSnSet *and* Spectrum *objects*

Description

The normalise method (also available as normalize) performs basic normalisation on spectra intensities of single spectra ("Spectrum" or "Spectrum2" objects), whole experiments ("MSnExp" objects) or quantified expression data ("MSnSet" objects).

Raw spectra and experiments are normalised using max or sum only. For MSMS spectra could be normalised to their precursor additionally. Each peak intensity is divided by the highest intensity in the spectrum, the sum of intensities or the intensity of the precursor. These methods aim at facilitating relative peaks heights between different spectra.

The method parameter for "MSnSet" can be one of sum, max, quantiles, center.mean, center.median, quantiles.robust or vsn. For sum and max, each feature's reporter intensity is divided by the maximum or the sum respectively. These two methods are applied along the features (rows). center.mean and center.median translate the respective sample (column) intensities according to the column mean or median. Using quantiles or quantiles.robust applies (robust) quantile normalisation, as implemented in normalize.quantiles and normalize.quantiles.robust of the preprocessCore package. vsn uses the vsn2 function from the vsn package. Note that the latter also glog-transforms the intensities. See respective manuals for more details and function arguments.

A scale method, mimicking the base scale method exists for "MSnSet" instances. See ?base::scale for details.

Arguments

object   An object of class "Spectrum", "Spectrum2", "MSnExp" or "MSnSet".

method   A character vector of length one that describes how to normalise the object. See description for details.

...   Additional arguments passed to the normalisation function.

Methods

The normalise methods:

signature(object = "MSnSet", method = "character") Normalises the object reporter ions intensities using method.

signature(object = "MSnExp", method = "character") Normalises the object peak intensities using method.

signature(object = "Spectrum", method = "character") Normalises the object peak intensities using method.

signature(object = "Spectrum2", method = "character", precursorIntensity) Normalises the object peak intensities using method. If method == "precursor", precursorIntensity allows to specify the intensity of the precursor manually.

The scale method:

signature(x = "MSnSet", center = "logical", scale = "logical") See ?base::scale.

Examples

```
## quantifying full experiment
data(msnset)
msnset.nrm <- normalise(msnset, "quantiles")
msnset.nrm
```

---

npcv   *Non-parametric coefficient of variation*

Description

Calculates a non-parametric version of the coefficient of variation where the standard deviation is replaced by the median absolute deviations (see mad for details) and divided by the absolute value of the mean.

Usage

```
npcv(x, na.rm = TRUE)
```

Arguments

x   A numeric.

na.rm   A logical (default is TRUE indicating whether NA values should be stripped before the computation of the median absolute deviation and mean.

Details

Note that the mad of a single value is 0 (as opposed to NA for the standard deviation, see example below).

Value

A numeric.

Author(s)

Laurent Gatto

Examples

```
set.seed(1)
npcv(rnorm(10))
replicate(10, npcv(rnorm(10)))
npcv(1)
mad(1)
sd(1)
```

---

nQuants   *Count the number of quantitfied features.*

Description

This function counts the number of quantified features, i.e non NA quantitation values, for each
group of features for all the samples in an ["MSnSet"](#) object. The group of features are defined by a
feature variable names, i.e the name of a column of fData(object).

Usage

```
nQuants(object, fcol)
```

Arguments

object   An instance of class ["MSnSet"](#).

fcol   The feature variable to consider when counting the number of quantified featues.

Details

This function is typically used after [topN](#) and before [combineFeatures](#), when the summerising
function is sum, or any function that does not normalise to the number of features aggregated. In
the former case, sums of feautres might be the result of 0 (if no feature was quantified) to n (if all
topN's n features were quantified) features, and one might want to rescale the sums based on the
number of non-NA features effectively summed.

Value

A matrix of dimensions length(levels(factor(fData(object)[, fcol]))) by ncol(object)
of integers.

Author(s)

Laurent Gatto

Examples

```
data(msnset)
n <- 2
msnset <- topN(msnset, groupBy = fData(msnset)$ProteinAccession, n)
m <- nQuants(msnset, fcol = "ProteinAccession")
msnset2 <- combineFeatures(msnset,
                           groupBy = fData(msnset)$ProteinAccession,
                           fun = sum)
stopifnot(dim(n) == dim(msnset2))
head(exprs(msnset2))
head(exprs(msnset2) * (n/m))
```

---

pickPeaks-methods   *Peak Detection for 'MSnExp' or 'Spectrum' instances*

Description

This method performs a peak picking on individual spectra (Spectrum instances) or whole experiments (MSnExp instances) to create centroided spectra. For noisy spectra there are currently two different noise estimators, the Median Absolute Deviation (method = "MAD") and Friedman's Super Smoother (method = "SuperSmoother"), as implemented in the MALDIquant::detectPeaks and MALDIquant::estimateNoise functions respectively.

Methods

signature(x = "MSnExp", halfWindowSize = "integer", method = "character", SNR = "numeric", verbose
    Performs the peak picking for all spectra in an MSnExp instance. method could be "MAD" or
    "SuperSmoother". halfWindowSize controls the window size of the peak picking algorithm.
    The resulting window size is 2 * halfWindowSize + 1. The size should be nearly (or slightly
    larger) the *FWHM* (full width at half maximum). A local maximum is considered as peak if
    its intensity is SNR times larger than the estimated noise. The arguments ... are passed to the
    noise estimator functions. Currenlty only the method = "SuperSmoother" accepts additional
    arguments, e.g. span. Please see [supsmu](#) for details. This method displays a progress bar if
    verbose = TRUE. Returns an MSnExp instance with centroided spectra.

signature(x = "Spectrum", method = "character", halfWindowSize = "integer", ...)
    Performs the peak picking for the spectrum (Spectrum instance). This method is the same as
    above but returns a centroided Spectrum instead of an MSnExp object. It has no verbose
    argument. Please read the details for the above MSnExp method.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

References

S. Gibb and K. Strimmer. 2012. MALDIquant: a versatile R package for the analysis of mass spectrometry data. Bioinformatics 28: 2270-2271. <http://strimmerlab.org/software/maldiquant/>

See Also

[clean](#), [removePeaks](#) [smooth](#) and [trimMz](#) for other spectra processing methods.

Examples

```
sp1 <- new("Spectrum1",
           intensity = c(1:6, 5:1),
           mz = 1:11)
sp2 <- pickPeaks(sp1)
intensity(sp2)
```

```
data(itraqdata)
itraqdata2 <- pickPeaks(itraqdata)
processingData(itraqdata2)
```

---

plot-methods   *Plotting 'Spectrum' object(s)*

Description

These method plot mass spectra MZ values against the intensities. Full spectra (using the full parameter) or specific peaks of interest can be plotted using the reporters parameter. If reporters are specified and full is set to 'TRUE', a sub-figure of the reporter ions is inlaid inside the full spectrum.

If an "MSnExp" is provided as argument, all the spectra are aligned vertically. Experiments can be subset to extract spectra of interest using the [operator or extractPrecSpectra methods.

The methods make use the ggplot2 system. An object of class 'ggplot' is returned invisibly.

If a single "Spectrum2" and a "character" representing a valid peptide sequence are passed as argument, the expected fragment ions are calculated and matched/annotated on the spectum plot.

Arguments

x	Objects of class "Spectrum", "Spectrum2" or "MSnExp" to be plotted.
y	Missing, "Spectrum" or "character".
reporters	An object of class "ReporterIons" that defines the peaks to be plotted. If not specified, full must be set to 'TRUE'.
full	Logical indicating whether full spectrum (respectively spectra) of only reporter ions of interest should be plotted. Default is 'FALSE', in which case reporters must be defined.
centroided.	Logical indicating if spectrum or spectra are in centroided mode, in which case peaks are plotted as histograms, rather than curves.
plot	Logical specifying whether plot should be printed to current device. Default is 'TRUE'.
w1	Width of sticks for full centroided spectra. Default is to use maximum MZ value divided by 500.
w2	Width of histogram bars for centroided reporter ions plots. Default is 0.01. See below for more details.

Methods

signature(x = "MSnExp", y = "missing", reporters = "ReporterIons", full = "logical", plot = "logica
    Plots all the spectra in the MSnExp object vertically. One of reporters must be defined or full
    set to 'TRUE'. In case of MSnExp objects, repoter ions are not inlaid when full is 'TRUE'.

signature(x = "Spectrum", y = "missing", reporters = "ReporterIons", full = "logical", centroided. =
Displays the MZs against intensities of the Spectrum object as a line plot. At least one of
reporters being defined or full set to 'TRUE' is required. reporters and full are used
only for *"Spectrum2"* objects. Full *"Spectrum1"* spectra are plotted by default.

signature(x = "Spectrum2", y = "character", orientation = "numeric", add = "logical", col = "charac
Plots a single MS2 spectrum and annotates the fragment ions based on the matching between
the peaks in x and the fragment peaks calculated from the peptide sequence y. The default val-
ues are orientation=1, add=FALSE, col="#74ADD1", pch=NA, xlab="m/z", ylab="intensity",
ylim=c(0, 1), tolerance=0.1, relative=FALSE, type=c("b", "y"), modifications=c(C=160.030649),
z=1, fragments=MSnbase:::calculateFragments_Spectrum2 and fragments.cex=0.75.
Additional arguments ... are passed to plot.default.

Author(s)

Laurent Gatto <lg390@cam.ac.uk> and Sebastian Gibb

See Also

calculateFragments to calculate ions produced by fragmentation and plot.Spectrum.Spectrum
to plot and compare 2 spectra and their shared peaks.

Examples

```
data(itraqdata)
## plotting experiments
plot(itraqdata[1:2], reporters = iTRAQ4)
plot(itraqdata[1:2], full = TRUE)
## plotting spectra
plot(itraqdata[[1]],reporters = iTRAQ4, full = TRUE)

itraqdata2 <- pickPeaks(itraqdata)
i <- 14
s <- as.character(fData(itraqdata2)[i, "PeptideSequence"])
plot(itraqdata2[[i]], s, main = s)
```

---

plot.Spectrum.Spectrum-methods
                    Plotting a 'Spectrum' vs another 'Spectrum' object.

Description

These method plot mass spectra MZ values against the intensities as line plots. The first spectrum
is plotted in the upper panel and the other in upside down in the lower panel. Common peaks are
drawn in a slightly darker colour. If a peptide sequence is provided it automatically calculates and
labels the fragments.

Arguments

x	Object of class "Spectrum" .
y	Object of class "Spectrum" .
...	Further arguments passed to internal functions.

Methods

signature(x = "Spectrum", y = "Spectrum", ...) Plots two spectra against each other. Common peaks are drawn in a slightly darker colour. The ... arguments are passed to the internal functions. Currently tolerance, relative, sequences and most of the plot.default arguments (like xlim, ylim, main, xlab, ylab, ...) are supported. You could change the tolerance (default 25e-6) and decide whether this tolerance should be applied relative (default relative = TRUE) or absolute (relative = FALSE) to find and colour common peaks. Use a character vector of length 2 to provide sequences which would be used to calculate and draw the corresponding fragments. If sequences are given the type argument (default: type=c("b", "y") specify the fragment types which should calculated. Also it is possible to allow some modifications. Therefore you have to apply a named character vector for modifications where the name corresponds to the one-letter-code of the modified amino acid (default: Carbamidomethyl modifications=c(C=160.030649)). See calculateFragments for details.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

See Also

More spectrum plotting available in plot.Spectrum.

Examples

```
## find path to a mzXML file
file <- dir(system.file(package = "MSnbase", dir = "extdata"),
            full.name = TRUE, pattern = "mzXML$")

## create basic MSnExp
msexp <- readMSData(file)

## centroid them
msexp <- pickPeaks(msexp)

## plot the first against the second spectrum
plot(msexp[[1]], msexp[[2]])

## add sequence information
plot(msexp[[1]], msexp[[2]], sequences=c("VESITARHGEVLQLRPK",
                                         "IDGQWVTHQWLKK"))


itraqdata2 <- pickPeaks(itraqdata)
```

```
(k <- which(fData(itraqdata2)[, "PeptideSequence"] == "TAGIQIVADDLTVTNPK"))
mzk <- precursorMz(itraqdata2)[k]
zk <- precursorCharge(itraqdata2)[k]
mzk * zk
plot(itraqdata2[[k[1]]], itraqdata2[[k[2]]])
```

---

plot2d-methods   *The 'plot2d' method for 'MSnExp' quality assessment*

Description

These methods plot the retention time vs. precursor MZ for the whole "MSnExp" experiment. Individual dots will be colour-coded to describe individual spectra's peaks count, total ion count, precursor charge (MS2 only) or file of origin.

The methods make use the ggplot2 system. An object of class 'ggplot' is returned invisibly.

Arguments

object   An object of class "MSnExp" or a data.frame. In the latter case, the data frame
                must have numerical columns named 'retention.time' and 'precursor.mz' and
                one of 'tic', 'file', 'peaks.count' or 'charge', depending on the z parameter. Such
                a data frame is typically generated using the header method on "MSnExp" object.

z   A character indicating according to what variable to colour the dots. One of, possibly abreviated, "ionCount" (total ion count), "file" (raw data file), "peaks.count"
                (peaks count) or "charge" (precursor charge).

alpha   Numeric [0,1] indicating transparence level of points.

plot   A logical indicating whether the plot should be printed (default is 'TRUE').

Methods

signature(object = "MSnExp", ...) Plots a 'MSnExp' summary.

signature(object = "data.frame", ...) Plots a summary of the 'MSnExp' experiment described by the data frame.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

The plotDensity and plotMzDelta methods for other QC plots.

Examples

```
itraqdata
plot2d(itraqdata,z="ionCount")
plot2d(itraqdata,z="peaks.count")
plot2d(itraqdata,z="charge")
```

---

plotDensity-methods *The 'plotDensity' method for 'MSnExp' quality assessment*

Description

These methods plot the distribution of several parameters of interest for the different precursor charges for *"MSnExp"* experiment.

The methods make use the ggplot2 system. An object of class 'ggplot' is returned invisibly.

Arguments

object   An object of class *"MSnExp"* or and 'data.frame'. In the latter case, the data frame must have numerical columns named 'charge' and one of 'precursor.mz', 'peaks.count' or 'ionCount', depending on the z parameter. Such a data frame is typically generated using the header method on *"MSnExp"* object.

z   A character indicating which parameter's densitiy to plot. One of, possibly abreviated, "ionCount" (total ion count), "peaks.count" (peaks count) or "precursor.mz" (precursor MZ).

log   Logical, whether to log transform the data (default is 'FALSE').

plot   A logical indicating whether the plot should be printed (default is 'TRUE').

Methods

signature(object = "MSnExp", ...) Plots a 'MSnExp' summary.

signature(object = "data.frame", ...) Plots a summary of the 'MSnExp' experiment described by the data frame.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

The plot2d and plotDensity methods for other QC plots.

Examples

```
itraqdata
plotDensity(itraqdata,z="ionCount")
plotDensity(itraqdata,z="peaks.count")
plotDensity(itraqdata,z="precursor.mz")
```

`plotMzDelta-methods`   *The delta m/z plot*

Description

The m/z delta plot illustrates the suitability of MS2 spectra for identification by plotting the m/z differences of the most intense peaks. The resulting histogram should optimally shown outstanding bars at amino acid residu masses. The plots have been described in Foster *et al* 2011.

Only a certain percentage of most intense MS2 peaks are taken into account to use the most significant signal. Default value is 10% (see `percentage` argument). The difference between peaks is then computed for all individual spectra and their distribution is plotted as a histogram where single bars represent 1 m/z differences. Delta m/z between 40 and 200 are plotted by default, to encompass the residue masses of all amino acids and several common contaminants, although this can be changes with the `xlim` argument.

In addition to the processing described above, isobaric reporter tag peaks (see the `reporters` argument) and the precursor peak (see the `precMz` argument) can also be removed from the MS2 spectrum, to avoid interence with the fragment peaks.

Note that figures in Foster *et al* 2011 have been produced and optimised for centroided data. Application of the plot as is for data in profile mode has not been tested thoroughly, although the example below suggest that it might work.

The methods make use the `ggplot2` system. An object of class `ggplot` is returned invisibly.

Most of the code for plotMzDelta has kindly been contributed by Guangchuang Yu.

Arguments

`object`	An object of class `MSnExp` or `mzRramp` (from the `mzR` package) containing MS2 spectra.
`reporters`	An object of class class `"ReporterIons"` that defines which reporter ion peaks to set to 0. The default value `NULL` leaves the spectra as they are.
`subset`	A numeric between 0 and 1 to use a subset of `object`'s MS2 spectra.
`percentage`	The percentage of most intense peaks to be used for the plot. Default is 0.1.
`precMz`	A numeric of length one or `NULL` default. In the latter (and preferred) case, the precursor m/z values are extracted from the individual MS2 spectra using the `precursorMz` method.
`precMzWidth`	A numeric of length 1 that specifies the width around the precursor m/z where peaks are set to 0. Default is 2.
`bw`	A numeric specifying the bandwith to be used to bin the delta m/z value to plot the histogram. Default if 1. See `geom_histogram` for more details.
`xlim`	A numeric of length 2 specifying the range of delta m/z to plot on the histogram. Default is `c(40,200)`.
`withLabels`	A logical defining if amino acid residue labels are plotted on the figure. Default is `TRUE`.

size   A numeric of length 1 specifying the font size of amino acids lables. Default is
                2.5.

plot   A logical of length 1 that defines whether the figure should be plotted on the
                active device. Default is TRUE. Note that the ggplot object is always returned
                invisibly.

verbose   A logical of length 1 specifying whether textual output and a progress bar
                illustration the progress of data processing should be printed. Default is TRUE

Methods

signature(object = "MSnExp", ...) Plots and (invisibly) returns the m/z delta histogram.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

Foster JM, Degroeve S, Gatto L, Visser M, Wang R, Griss J, Apweiler R, Martens L. "A pos-
teriori quality control for the curation and reuse of public proteomics data." *Proteomics*, 2011
Jun;11(11):2182-94. doi:10.1002/pmic.201000602. Epub 2011 May 2. PMID: 21538885

See Also

The plotDensity and plot2d methods for other QC plots.

Examples

```
mzdplot <- plotMzDelta(itraqdata,
                       subset = 0.5,
                       reporters = iTRAQ4,
                       verbose = FALSE, plot = FALSE)
## let's retrieve peptide sequence information
## and get a table of amino acids
peps <- as.character(fData(itraqdata)$PeptideSequence)
aas <- unlist(strsplit(peps,""))
## table of aas
table(aas)
## mzDelta plot
print(mzdplot)
```

plotNA-methods   *Exploring missing data in 'MSnSet' instances*

Description

These methods produce plots that illustrate missing data.

is.na returns the expression matrix of it MSnSet argument as a matrix of logicals referring whether the corresponding cells are NA or not. It is generally used in conjunction with table and image (see example below).

The plotNA method produces plots that illustrate missing data. The completeness of the full dataset or a set of proteins (ordered by increasing NA content along the x axis) is represented. The methods make use the ggplot2 system. An object of class 'ggplot' is returned invisibly.

Methods

is.na signature(x = "MSnSet") Returns the a matrix of logicals of dimensions dim(x) specifiy-ing if respective values are missing in the MSnSet's expression matrix.

plotNA signature(object = "MSnSet", pNA = "numeric") Plots missing data for an MSnSet instance. pNA is a numeric of length 1 that specifies the percentage of accepted missing data values per features. This value will be highlighted with a point on the figure, illustrating the overall percentage of NA values in the full data set and the number of proteins retained. Default is 1/2.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

See also the [filterNA](#) method to filter out features with a specified proportion if missing values.

Examples

```
data(msnset)
exprs(msnset)[sample(prod(dim(msnset)), 120)] <- NA

head(is.na(msnset))
table(is.na(msnset))
image(msnset)

plotNA(msnset, pNA = 1/4)
```

---

precSelection   *Number of precursor selection events*

Description

precSelection computes the number of selection events each precursor ions has undergone in an tandem MS experiment. This will be a function of amount of peptide loaded, chromatography efficiency, exclusion time,... and is useful when optimising and experimental setup. This function returns a named integer vector or length equal to the number of unique precursor MZ values in the original experiment. See n parameter to set the number of MZ significant decimals.

precSelectionTable is a wrapper around precSelection and returns a table with the number of single, 2-fold, ... selection events.

Usage

```
precSelection(object,n)
```

Arguments

object   An instane of class ``MSnExp``.

n   The number of decimal places to round the precursor MZ to. Is passed to the
                round function.

Value

A named integer in case of precSelection and a table for precSelectionTable.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

```
precSelection(itraqdata)
precSelection(itraqdata,n=2)
precSelectionTable(itraqdata)
## only single selection event in this reduced exeriment
```

---

pSet-class   *Class to Contain Raw Mass-Spectrometry Assays and Experimental*
                            Metadata

Description

Container for high-throughput mass-spectrometry assays and experimental metadata. This class is based on Biobase's ``eSet`` virtual class, with the notable exception that 'assayData' slot is an environment contain objects of class ``Spectrum``.

Objects from the Class

A virtual Class: No objects may be created from it. See ``MSnExp`` for instantiatable sub-classes.

Slots

assayData: Object of class "environment" containing the MS spectra (see "Spectrum1" and "Spectrum2"). Slot is inherited from "pSet".

phenoData: Object of class "AnnotatedDataFrame" containing experimenter-supplied variables describing sample (i.e the individual tags for an labelled MS experiment) See phenoData for more details. Slot is inherited from "pSet".

featureData: Object of class "AnnotatedDataFrame" containing variables describing features (spectra in our case), e.g. identificaiton data, peptide sequence, identification score,... (inherited from "eSet"). See featureData for more details. Slot is inherited from "pSet".

experimentData: Object of class "MIAPE", containing details of experimental methods. See experimentData for more details. Slot is inherited from "pSet".

protocolData: Object of class "AnnotatedDataFrame" containing equipment-generated variables (inherited from "eSet"). See protocolData for more details. Slot is inherited from "pSet".

processingData: Object of class "MSnProcess" that records all processing. Slot is inherited from "pSet".

.cache: Object of class environment used to cache data. Under development.

.__classVersion__: Object of class "Versions" describing the versions of the class.

Extends

Class "VersionedBiobase", directly. Class "Versioned", by class "VersionedBiobase", distance 2.

Methods

Methods defined in derived classes may override the methods described here.

[signature(x = "pSet"): Subset current object and return object of same class.

[[signature(x = "pSet"): Direct access to individual spectra.

abstract Access abstract in experimentData.

assayData signature(object = "pSet"): Access the assayData slot. Returns an environment.

desciption signature(x = "pSet"): Synonymous with experimentData.

dim signature(x = "pSet"): Returns the dimensions of the phenoData slot.

experimentData signature(x = "pSet"): Access details of experimental methods.

featureData signature(x = "pSet"): Access the featureData slot.

fData signature(x = "pSet"): Access feature data information.

featureNames signature(x = "pSet"): Coordinate access of feature names (e.g spectra, peptides or proteins) in assayData slot.

fileNames signature(object = "pSet"): Access file names in the processingData slot.

fromFile signature(object = "pSet"): Access raw data file indexes (to be found in the 'code-processingData' slot) from which the individual object's spectra where read from.

centroided signature(object = "pSet"): Indicates whether individual spectra are centroided ('TRUE') of uncentroided ('FALSE'). Use centroided(object) <- value to update a whole experiment, ensuring that object and value have the same length.

fvarMetadata signature(x = "pSet"): Access metadata describing features reported in fData.

fvarLabels signature(x = "pSet"): Access variable labels in featureData.

length signature(x = "pSet"): Returns the number of features in the assayData slot.

notes signature(x = "pSet"): Retrieve and unstructured notes associated with pSet in the experimentData slot.

pData signature(x = "pSet"): Access sample data information.

phenoData signature(x = "pSet"): Access the phenoData slot.

processingData signature(object = "pSet"): Access the processingData slot.

protocolData signature(x = "pSet"): Access the protocolData slot.

pubMedIds signature(x = "pSet"): Access PMIDs in experimentData.

sampleNames signature(x = "pSet"): Access sample names in phenoData.

spectra signature(x = "pSet", ...): Access the assayData slot, returning the features as a list. Additional arguments are currently ignored.

varMetadata signature(x = "pSet"): Access metadata describing variables reported in pData.

varLabels signature(x = "pSet"): Access variable labels in phenoData.

acquisitionNum signature(object = "pSet"): Accessor for spectra acquisition numbers.

scanIndex signature(object = "pSet"): Accessor for spectra scan indices.

collisionEnergy signature(object = "pSet"): Accessor for MS2 spectra collision energies.

intensity signature(object = "pSet", ...): Accessor for spectra instenities, returned as named list. Additional arguments are currently ignored.

msInfo signature(object = "pSet"): Prints the MIAPE-MS meta-data stored in the experimentData slot.

msLevel signature(object = "pSet"): Accessor for spectra MS levels.

mz signature(object = "pSet", ...): Accessor for spectra M/Z values, returned as a named list. Additional arguments are currently ignored.

peaksCount signature(object = "pSet"): Accessor for spectra preak counts.

peaksCount signature(object = "pSet", scans = "numeric"): Accessor to scans spectra preak counts.

polarity signature(object = "pSet"): Accessor for MS1 spectra polarities.

precursorCharge signature(object = "pSet"): Accessor for MS2 precursor charges.

precursorIntensity signature(object = "pSet"): Accessor for MS2 precursor intensity.

precursorMz signature(object = "pSet"): Accessor for MS2 precursor M/Z values.

precAcquisitionNum signature(object = "pSet"): Accessor for MS2 precursor scan numbers.

precScanNum see precAcquisitionNum.

rtime signature(object = "pSet", ...): Accessor for spectra retention times. Additional arguments are currently ignored.

tic signature(object = "pSet", ...): Accessor for spectra total ion counts. Additional arguments are currently ignored.

ionCount signature(object = "pSet"): Accessor for spectra total ion current.

header signature(object = "pSet"): Returns a data frame containing all available spectra parameters (MSn only).

header signature(object = "pSet", scans = "numeric"): Returns a data frame containing scans spectra parameters (MSn only).

Additional accessors for the experimental metadata (experimentData slot) are defined. See "MIAPE" for details.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

The "eSet" class, on which pSet is based.

See Also

"MSnExp" for an instantiatable application of pSet.

Examples

showClass("pSet")

---

purityCorrect-methods *Performs reporter ions purity correction*

Description

Manufacturers sometimes provide purity correction values indicating the percentages of each reporter ion that have masses differing by +/- n Da from the nominal reporter ion mass due to isotopic variants. This correction is generally applied after reporter peaks quantitation.

Purity correction here is applied using solve from the base package using the purity correction values as coefficient of the linear system and the reporter quantities as the right-hand side of the linear system. 'NA' values are ignored and negative intensities after correction are also set to 'NA'.

A more elaborated purity correction method is described in Shadforth *et al.*, i-Tracker: for quantitative proteomics using iTRAQ. BMC Genomics. 2005 Oct 20;6:145. (PMID 16242023).

Function makeImpuritiesMatrix(x, filename, edit = TRUE) helps the user to create such a matrix. The function can be used in two ways. If given an integer x, it is used as the dimension of the square matrix (i.e the number of reporter ions). For TMT6-plex and iTRAQ4-plex, default values taken from manufacturer's certification sheets are used as templates, but batch specific values should be used whenever possible. Alternatively, the filename of a csv spreadsheet can be provided. The sheet should define the correction factors as illustrated below (including

reporter names in the first column and header row) and the corresponding correction matrix is calculated. Examples of such csv files are available in the package's extdata directory. Use dir(system.file("extdata", package = "MSnbase"), pattern =   "PurityCorrection", full.names = TRUE) to locate them. If edit = TRUE, the the matrix can be edited before it is returned.

Arguments

object   An object of class `"MSnSet"`.

impurities   A square 'matrix' of dim equal to ncol(object) defining the correction coeffi-
                cients to be applied. The reporter ions should be ordered along the columns and
                the relative percentages along the rows.

                As an example, below is the correction factors as provided in an ABI iTRAQ
                4-plex certificate of analysis:

reporter	% of -2	% of -1	% of +1	% of +2
114	0.0	1.0	5.9	0.2
115	0.0	2.0	5.6	0.1
116	0.0	3.0	4.5	0.1
117	0.1	4.0	3.5	0.1

                The impurity table will be

0.929	0.059	0.002	0.000
0.020	0.923	0.056	0.001
0.000	0.030	0.924	0.045
0.000	0.001	0.040	0.923

                where, the diagonal is computed as 100 - sum of rows of the original table and
                subsequent cells are directly filled in.

                Similarly, for TMT 6-plex tags, we observe

reporter	% of -3	% of -2	% of -1	% of +1 %	% of +2	% of +3
126	0	0	0	6.1	0	0
127	0	0	0.5	6.7	0	0
128	0	0	1.1	4.2	0	0
129	0	0	1.7	4.1	0	0
130	0	0	1.6	2.1	0	0
131	0	0.2	3.2	2.8	0	0

                and obtain the following impurity correction matrix

0.939	0.061	0.000	0.000	0.000	0.000
0.005	0.928	0.067	0.000	0.000	0.000
0.000	0.011	0.947	0.042	0.000	0.000
0.000	0.000	0.017	0.942	0.041	0.000

```
0.000   0.000   0.000   0.016   0.963   0.021
0.000   0.000   0.000   0.002   0.032   0.938
```

For iTRAQ 8-plex, given the following correction factors (to make such a matrix square, if suffices to add -4, -3, +3 and +4 columns filled with zeros):

TAG	-2	-1	+1	+2
113	0	2.5	3	0.1
114	0	1	5.9	0.2
115	0	2	5.6	0.1
116	0	3	4.5	0.1
117	0.1	4	3.5	0.1
118	0.1	2	3	0.1
119	0.1	2	4	0.1
121	0.1	2	3	0.1

we calculate the impurity correction matrix shown below

	113	114	115	116	117	118	119	121
% reporter 113	0.944	0.030	0.001	0.000	0.000	0.000	0.000	0.000
% reporter 114	0.010	0.929	0.059	0.002	0.000	0.000	0.000	0.000
% reporter 115	0.000	0.020	0.923	0.056	0.001	0.000	0.000	0.000
% reporter 116	0.000	0.000	0.030	0.924	0.045	0.001	0.000	0.000
% reporter 117	0.000	0.000	0.001	0.040	0.923	0.035	0.001	0.000
% reporter 118	0.000	0.000	0.000	0.001	0.020	0.948	0.030	0.001
% reporter 119	0.000	0.000	0.000	0.000	0.001	0.020	0.938	0.040
% reporter 121	0.000	0.000	0.000	0.000	0.000	0.001	0.020	0.948

Finally, for a TMT 10-plex impurity matrix

.	-2	-1	1	2
126	0.0	0.00	4.69 (127C)	0.0 (128N)
127N	0.0	0.40	6.50 (128N)	0.0 (128C)
127C	0.0	0.20 (126)	4.60 (128C)	0.3 (129N)
128N	0.0	0.90 (127N)	4.70 (129N)	0.2 (129C)
128C	0.1 (126)	0.53 (127C)	2.59 (129C)	0.0 (130N)
129N	0.0 (127N)	0.73 (128N)	2.49 (130N)	0.0 (130C)
129C	0.0 (127C)	1.30 (128C)	2.50 (130C)	0.0 (131)
130N	0.0 (128N)	1.20 (129N)	2.80 (131)	2.7
130C	0.1 (128C)	2.90 (129C)	2.90	0.0
131	0.0 (129N)	2.36 (130N)	1.43	0.0

the impurity correction matrix is

.	126	127N	127C	128N	128C	129N	129C	130N	130C	131
% reporter 126	0.9531	0.000	0.0469	0.0000	0.0000	0.0000	0.0000	0.0000	0.000	0.0000
% reporter 127N	0.0000	0.931	0.0000	0.0650	0.0000	0.0000	0.0000	0.0000	0.000	0.0000
% reporter 127C	0.0020	0.000	0.9490	0.0000	0.0460	0.0030	0.0000	0.0000	0.000	0.0000
% reporter 128N	0.0000	0.009	0.0000	0.9420	0.0000	0.0470	0.0020	0.0000	0.000	0.0000
% reporter 128C	0.0010	0.000	0.0053	0.0000	0.9678	0.0000	0.0259	0.0000	0.000	0.0000
% reporter 129N	0.0000	0.000	0.0000	0.0073	0.0000	0.9678	0.0000	0.0249	0.000	0.0000
% reporter 129C	0.0000	0.000	0.0000	0.0000	0.0130	0.0000	0.9620	0.0000	0.025	0.0000
% reporter 130N	0.0000	0.000	0.0000	0.0000	0.0000	0.0120	0.0000	0.9330	0.000	0.0280
% reporter 130C	0.0000	0.000	0.0000	0.0000	0.0010	0.0000	0.0290	0.0000	0.941	0.0000
% reporter 131	0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0236	0.000	0.9621

These examples are provided as defaults impurity correction matrices in makeImpuritiesMatrix.

Methods

```
signature(object = "MSnSet", impurities = "matrix")
```

Examples

```
## quantifying full experiment
data(msnset)
impurities <- matrix(c(0.929,0.059,0.002,0.000,
                       0.020,0.923,0.056,0.001,
                       0.000,0.030,0.924,0.045,
                       0.000,0.001,0.040,0.923),
                     nrow=4, byrow = TRUE)
## or, using makeImpuritiesMatrix()
## Not run: impurities <- makeImpuritiesMatrix(4)
msnset.crct <- purityCorrect(msnset, impurities)
head(exprs(msnset))
head(exprs(msnset.crct))
processingData(msnset.crct)

## default impurity matrix for iTRAQ 8-plex
makeImpuritiesMatrix(8, edit = FALSE)

## default impurity matrix for TMT 10-plex
makeImpuritiesMatrix(10, edit = FALSE)
```

---

quantify-methods   *Quantifies 'MSnExp' and 'Spectrum' objects*

Description

This method quantifies individual "Spectrum" objects or full "MSnExp" experiments. Current, MS2-level isobar tagging using iTRAQ and TMT (or any arbitrary peaks of interest, see "ReporterIons")

and MS2-level label-free quantitation (spectral counting, spectral index or spectral abundance factor) are available.

Isobaric tag peaks of single spectra or complete experiments can be quantified using appropriate methods. Label-free quantitation is available only for MSnExp experiments.

Since version 1.13.5, parallel quantitation is supported by the BiocParallel package and controlled by the BPPARAM argument.

Arguments

object	An instance of class ["Spectrum"](#) (isobaric tagging only) or ["MSnExp"](#).
method	Peak quantitation method. For isobaric tags, one of, possibly abreviated "trapezoidation", "max", or "sum". These methods return respectively the area under the peak(s), the maximum of the peak(s) or the sum of all intensities of the peak(s).
	For label-free quantitation, one of "SI" (spectral index), "SIgi" (global intensity spectral index), "SIn" (normalised spectral index), "SAF" (spectral abundance factor) or "NSAF" (normalised spectral abundance factor).
	Finally, the simple "count" method counts the occurrence of the respective spectra (at this stage all 1s) that can then be used as input to [combineFeatures](#) to implement spectra counting.
reporters	An instance of class ["ReporterIons"](#) that defines the peak(s) to be quantified. For isobaric tagging only.
strict	For isobaric tagging only. If strict is FALSE (default), the quantitation is performed using data points along the entire width of a peak. If strict is set to TRUE, once the apex(es) is/are identified, only data points within apex +/- width of reporter (see ["ReporterIons"](#)) are used for quantitation.
BPPARAM	Support for parallel processing using the BiocParallel infrastructure. When missing (default), the default registered BiocParallelParam parameters are applied using bpparam(). Alternatively, one can pass a valid BiocParallelParam parameter instance: SnowParam, MulticoreParam, DoparParam, ... see the BiocParallel package for details.
parallel	Deprecated. Please see BPPARAM.
qual	Should the qual slot be populated. Default is TRUE.
verbose	Verbose of the output (only for MSnExp objects).
...	Further arguments passed to the quantitation functions.

Details

["ReporterIons"](#) define specific MZ at which peaks are expected and a window around that MZ value. A peak of interest is searched for in that window. Since version 1.1.2, warnings are not thrown anymore in case no data is found in that region or if the peak extends outside the window. This can be checked manually after quantitation, by inspecting the quantitation data (using the exprs accessor) for NA values or by comaring the lowerMz and upperMz columns in the ["MSnSet"](#) qual slot against the respective expected mz(reporters) +/- width(reporters).

Once the range of the curve is found, quantification is performed. If no data points are found in the expected region, NA is returned for the reporter peak MZ.

Note that for label-free, spectra that have not been identified (the corresponding fields in the feature data are populated with NA values) or that have been uniquely assigned to a protein (the nprot feature data is greater that 1) are removed prior to quantitation. The latter does not apply for method = "count" but can be applied manually with removeMultipleAssignment.

Methods

signature(object = "MSnExp", method = "character", reporters = "ReporterIons", verbose = "logical",

For isobaric tagging, quantifies peaks defined in reporters using method in all spectra of the MSnExp object. If verbose is set to TRUE, a progress bar will be displayed.

For label-free quantitation, the respective quantitation methods and normalisations are applied to the spectra. These methods require two additional arguments (...), namely the protein accession of identifiers (fcol, with detault value "accession") and the protein lengths (plength, with default value "length"). These values are available of the identification data had been collated using addIdentificationData.

An object of class "MSnSet" is returned containing the quantified feature expression and all meta data inherited from the MSnExp object argument.

signature(object = "Spectrum", method = "character", reporters = "ReporterIons")

Quantifies peaks defined in reporters using method in the Spectrum object (isobaric tagging only).

A list of length 2 will be returned. The first element, named peakQuant, is a 'numeric' of length equal to length(reporters) with quantitation of the reporter peaks using method.

The second element, names curveStats, is a 'data.frame' of dimension length(reporters) times 7 giving, for each reporter curve parameters: maximum intensity ('maxInt'), number of maxima ('nMaxInt'), number of data points defined the curve ('baseLength'), lower and upper MZ values for the curve ('lowerMz' and 'upperMz'), reporter ('reporter') and precursor MZ value ('precursor') when available.

Author(s)

Laurent Gatto <lg390@cam.ac.uk> and Sebastian Gibb <mail@sebastiangibb.de>

References

For details about the spectral index (SI), see Griffin NM, Yu J, Long F, Oh P, Shore S, Li Y, Koziol JA, Schnitzer JE. *Label-free, normalized quantification of complex mass spectrometry data for proteomic analysis*. Nat Biotechnol. 2010 Jan;28(1):83-9. doi: 10.1038/nbt.1592. PMID: 20010810; PubMed Central PMCID: PMC2805705.

For details about the spectra abundance factor, see Paoletti AC, Parmely TJ, Tomomori-Sato C, Sato S, Zhu D, Conaway RC, Conaway JW, Florens L, Washburn MP. *Quantitative proteomic analysis of distinct mammalian Mediator complexes using normalized spectral abundance factors*. PNAS. 2006 Dec 12;103(50):18928-33. PMID: 17138671; PubMed Central PMCID: PMC1672612.

Examples

```
## Quantifying a full experiment using iTRAQ4-plex tagging
data(itraqdata)
```

```
msnset <- quantify(itraqdata, method = "trap", reporters = iTRAQ4)
msnset

## specifying a custom parallel framework
## bp <- MulticoreParam(2L) # on Linux/OSX
## bp <- SnowParam(2L) # on Windows
## quantify(itraqdata[1:10], method = "trap", iTRAQ4, BPPARAM = bp)

## Checking for non-quantified peaks
sum(is.na(exprs(msnset)))

## Quantifying a single spectrum
qty <- quantify(itraqdata[[1]], method = "trap", iTRAQ4[1])
qty$peakQuant
qty$curveStats


## Label-free quantitation
## Raw (mzXML) and identification (mzid) files
quantFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "mzXML$")
identFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "dummyiTRAQ.mzid")

msexp <- readMSData(quantFile)
msexp <- addIdentificationData(msexp, identFile)
fData(msexp)$accession

si <- quantify(msexp, method = "SIn")
processingData(si)
exprs(si)

saf <- quantify(msexp, method = "NSAF")
processingData(saf)
exprs(saf)
```

---

readIspyData	*Reads an ispy2 result spread sheet and creates a fully featured 'MSnSet' instance.*

Description

Reads an ispy2 tab-delimited spreadsheet and generates the corresponding [MSnSet](#) object.

Usage

```
readIspyData(file = "ispy_results.tsv", uniquePeps = TRUE, pep = 0.05,
  na.rm = TRUE, min.int = 0, reporters = 19:23, keepAll = FALSE,
  verbose = TRUE)
```

Arguments

file	A `character`, indicating the file name to be read in. Default is "ispy_results.tsv".
uniquePeps	A `logical`, indicating whether only unique peptides should be included. Default is TRUE.
pep	A `numeric` indicating the posterior error probability threshold for peptides to be considered correctly identified. Default is 0.05.
na.rm	A `logical` indicating whether reporter ions containing one or more NA values should be excluded. Default is TRUE.
min.int	A `numeric` indicating the minimal summed intensity threshold for reporter data to be imported. Default is 0. Note that 'NA' values are excluded when summing the values.
reporters	A `numeric` indicating column indices of reporter ions quantitation data. Default is `19:23` for iTRAQ 4-plex.
keepAll	A `logical` that defines whether all features of the ispy result should be imported. If 'TRUE', 'pep', 'na.rm' and 'min.int' are ignored. This is equivalent to 'pep=1', 'na.rm=FALSE' and 'min.int=0'. Default is 'FALSE'.
verbose	A `logical` indicating whether verbose output is to be printed out.

Value

An object of class `"MSnSet"`.

Author(s)

Laurent Gatto

References

Ispy is a set of perl script to analyse SILAC, 15N and MSMS data developed by Phil D. Charles <pdc35@cam.ac.uk> at CCP http://www.bio.cam.ac.uk/proteomics/. No ispy references published yet.

See Also

readMSData to import raw data.

Examples

```
## Not run: ispy <- readIspyData("ispy_results.tsv")
```

---

readMgfData *Import mgf files as 'MSnExp' instances.*

Description

Reads a mgf file and generates an ["MSnExp"](#) object.

Usage

```
readMgfData(file, pdata = NULL, centroided = TRUE, smoothed = FALSE,
verbose = TRUE, cache = 1)
```

Arguments

file	character vector with file name to be read.
pdata	an object of class *"NAnnotatedDataFrame"*.
smoothed	Logical indicating whether spectra already smoothed or not. Default is 'FALSE'. Used to initialise *"MSnProcess"* object in processingData slot.
centroided	Logical indicating whether spectra are centroided or not. Default is 'TRUE'. Used to initialise *"MSnProcess"* object in processingData slot.
cache	Numeric indicating caching level. Default is 1. Under development.
verbose	verbosity flag.

Details

Note that when reading an mgf file, the original order of the spectra is lost. Thus, if the data was originally written to mgf from an MSnExp object using writeMgfData, although the feature names will be identical, the spectra are not as a result of the reordering. See example below.

Value

An instance of

Author(s)

Guangchuang Yu <guangchuangyu@gmail.com> and Laurent Gatto <lg390@cam.ac.uk>

See Also

[writeMgfData](#) method to write the content of *"Spectrum"* or *"MSnExp"* objects to mgf files. Raw data files can also be read with the [readMSData](#) function.

Examples

```
data(itraqdata)
writeMgfData(itraqdata, con="itraqdata.mgf", COM="MSnbase itraqdata")
itraqdata2 <- readMgfData("itraqdata.mgf")
## note that the order of the spectra is altered
## and precision of some values (precursorMz for instance)
match(signif(precursorMz(itraqdata2),4),signif(precursorMz(itraqdata),4))
## [1]  1 10 11 12 13 14 15 16 17 18 ...
## ... but all the precursors are there
all.equal(sort(precursorMz(itraqdata2)),
          sort(precursorMz(itraqdata)),
          check.attributes=FALSE,
          tolerance=10e-5)
## is TRUE
all.equal(as.data.frame(itraqdata2[[1]]),as.data.frame(itraqdata[[1]]))
## is TRUE
all.equal(as.data.frame(itraqdata2[[3]]),as.data.frame(itraqdata[[11]]))
## is TRUE
file <- dir(system.file(package="MSnbase",dir="extdata"),
            full.name=TRUE,
            pattern="test.mgf")
(x <- readMgfData(file))
x[[2]]
precursorMz(x[[2]])
precursorIntensity(x[[2]])
precursorMz(x[[1]])
precursorIntensity(x[[1]]) ## was not in test.mgf
scanIndex(x)
```

---

readMSData   *Imports mass-spectrometry raw data files as 'MSnExp' instances.*

Description

Reads as set of XML-based mass-spectrometry data files and generates an *"MSnExp"* object. This
function uses the functionality provided by the mzR package to access data and meta data in mzData,
mzXML and mzML.

Usage

```
readMSData(files, pdata = NULL, msLevel = 2, verbose = TRUE,
centroided = FALSE, smoothed = FALSE, removePeaks = 0, clean = FALSE,
cache = 1)
```

Arguments

files   character vector with file names to be read.

pdata   an object of class *"NAnnotatedDataFrame"*.

msLevel	MS level spectra to be read. Use '1' for MS1 spectra or any larger numeric for MSn spectra. Default is '2'.
centroided	Logical indicating whether spectra are centroided or not. Default is 'FALSE'. Used to initialise *"MSnProcess"* object in processingData slot.
smoothed	Logical indicating whether spectra already smoothed or not. Default is 'FALSE'. Used to initialise *"MSnProcess"* object in processingData slot.
removePeaks	If > 0 (default), all peaks less or equal then value will set to 0. See removePeaks for more details and examples.
clean	Logical indicating whether 0 intensity peaks should be discarded from spectra. Useful is removePeaks is set. Default is 'FALSE'. See clean for more details and examples.
cache	Numeric indicating caching level. Default is 0 for MS1 and 1 MS2 (or higher). Under development.
verbose	verbosity flag.

Value

An *"MSnExp"* object.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

"MSnExp" or readMgfData to read mgf peak lists.

Examples

```
file <- dir(system.file(package="MSnbase",dir="extdata"),
            full.name=TRUE,
            pattern="mzXML$")
aa <- readMSData(file)
aa
```

---

readMSnSet   *Read 'MSnSet'*

Description

This function reads data files to generate an MSnSet instance. It is a wrapper around Biobase's readExpressionSet function with an additional featureDataFile parameter to include feature data. See also readExpressionSet for more details. readMSnSet2 is a simple version that takes a single text spreadsheet as input and extracts the expression data and feature meta-data to create and MSnSet.

Usage

```
readMSnSet(exprsFile,
           phenoDataFile,
           featureDataFile,
           experimentDataFile,
           notesFile,
           path, annotation,
        exprsArgs = list(sep = sep, header = header, row.names = row.names, quote = quote, ...),
        phenoDataArgs = list(sep = sep, header = header, row.names = row.names, quote = quote, string
        featureDataArgs = list(sep = sep, header = header, row.names = row.names, quote = quote, stri
        experimentDataArgs = list(sep = sep, header = header, row.names = row.names, quote = quote, s
           sep = "\t",
           header = TRUE,
           quote = "",
           stringsAsFactors = FALSE,
           row.names = 1L,
           widget = getOption("BioC")$Base$use.widgets, ...)

readMSnSet2(file, ecol, fnames, ...)
```

Arguments

	Arguments direclty passed to readExpressionSet. The description is from the readExpressionSet documentation page.
	(character) File or connection from which to read expression values. The file should contain a matrix with rows as features and columns as samples. [read.table](#) is called with this as its file argument and further arguments given by exprsArgs.
phenoDataFile	(character) File or connection from which to read phenotypic data. [read.AnnotatedDataFrame](#) is called with this as its file argument and further arguments given by phenoDataArgs.
experimentDataFile	
	(character) File or connection from which to read experiment data. [read.MIAME](#) is called with this as its file argument and further arguments given by experimentDataArgs.
notesFile	(character) File or connection from which to read notes; [readLines](#) is used to input the file.
path	(optional) directory in which to find all the above files.
annotation	(character) A single character string indicating the annotation associated with this ExpressionSet.
exprsArgs	A list of arguments to be used with [read.table](#) when reading in the expression matrix.
phenoDataArgs	A list of arguments to be used (with [read.AnnotatedDataFrame](#)) when reading the phenotypic data.
experimentDataArgs	
	A list of arguments to be used (with [read.MIAME](#)) when reading the experiment data.
sep, header, quote, stringsAsFactors, row.names	
	arguments used by the [read.table](#)-like functions.

widget	A boolean value indicating whether widgets can be used. Widgets are NOT yet implemented for read.AnnotatedDataFrame.
...	Further arguments that can be passed on to the [read.table](#)-like functions.

Additional argument, specific to readMSnSet:

featureDataFile	
	(character) File or connection from which to read feature data. [read.AnnotatedDataFrame](#) is called with this as its file argument and further arguments given by phenoDataArgs.
featureDataArgs	
	A list of arguments to be used (with [read.AnnotatedDataFrame](#)) when reading the phenotypic data.

Arguments for readMSnSet2:

file	A character indicating the spreadsheet file. Default is to read the file as a comma-separated values (csv). If different, use the additional arguments, passed to [read.csv](#), to parametrise file import.
ecol	A numeric indicating the indices of the columns to be used as expression values. Can also be a character indicating the names of the columns. Caution must be taken of the column names are composed of special characters like '(' or '-' that will be converted to a '.'. If ecol does not match, the error message will dispay the column names are see by R.
fnames	An optional character indicating the column to be used as feature names.

Value

An instance of the [MSnSet](#) class.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

The [grepEcols](#) and [getEcols](#) helper functions to identify the ecol values. The MSnbase-io vignette illustrates these functions in detail. It can accessed with vignette("MSnbase-io").

Examples

```
## Not run:
exprsFile <- "path_to_intensity_file.csv"
fdatafile <- "path_to_featuredata_file.csv"
pdatafile <- "path_to_sampledata_file.csv"
## Read ExpressionSet with appropriate parameters
res <- readMSnSet(exprsFile, pdataFile, fdataFile, sep = "\t", header=TRUE)

## End(Not run)
```

---

readMzTabData   *Read an 'mzTab' file*

Description

This function can be used to create an ["MSnSet"](#) by reading and parsing an mzTab file. The metadata section is always used to populate the MSnSet's experimentData()@other$mzTab slot.

Usage

```
readMzTabData(file, what = c("PRT", "PEP", "PSM"), version = c("1.0",
  "0.9"), verbose = TRUE)
```

Arguments

file   A character with the mzTab file to be read in.

what   One of "PRT", "PEP" or "PSM", defining which of protein, peptide PSMs section should be returned as an MSnSet.

version   A character defining the format specification version of the mzTab file. Default is "1.0". Version "0.9" is available of backwards compatibility. See [readMzTabData_v0.9](#) for details.

verbose   Produce verbose output.

Value

An instance of class MSnSet.

Author(s)

Laurent Gatto

See Also

See [MzTab](#) and [MSnSetList](#) for details about the inners of readMzTabData.

Examples

```
testfile <- "https://mztab.googlecode.com/svn/examples//PRIDE_Exp_Complete_Ac_16649.xml-mztab.txt"
prot <- readMzTabData(testfile, "PRT")
prot
head(fData(prot))
head(exprs(prot))
psms <- readMzTabData(testfile, "PSM")
psms
head(fData(psms))
```

---

readMzTabData_v0.9 *Read an 'mzTab' file*

Description

This function can be used to create a "[MSnSet](#)" by reading and parsing an mzTab file. The metadata section is always used to populate the MSnSet's experimentData slot.

Usage

```
readMzTabData_v0.9(file, what = c("PRT", "PEP"), verbose = TRUE)
```

Arguments

file   A character with the mzTab file to be read in.

what   One of "PRT" or "PEP", defining which of protein of peptide section should be parse. The metadata section, when available, is always used to populate the experimentData slot.

verbose   Produce verbose output.

Value

An instance of class MSnSet.

Author(s)

Laurent Gatto

See Also

[writeMzTabData](#) to save an "[MSnSet](#)" as an mzTab file.

Examples

```
testfile <- "http://mztab.googlecode.com/svn/legacy/jmztab-1.0/examples/mztab_itraq_example.txt"
prot <- readMzTabData_v0.9(testfile, "PRT")
prot
pep <- readMzTabData_v0.9(testfile, "PEP")
pep
```

---

removeNoId-methods   *Removes non-identified features*

Description

The method removes non-identifed features in MSnExp and MSnSet instances using relevant information from the feaureData slot of a user-provide filtering vector of logicals.

Methods

signature(object = "MSnExp", fcol = "pepseq", keep = NULL) Removes the feature from
    object that have a feature fcol (default is "pepseq") equal to NA. Alternatively, one can also
    manually define keep, a vector of logical, defining the feature to be retained.

signature(object = "MSnSet", fcol = "pepseq", keep = NULL) As above of MSnSet in-
    stances.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

MSnExp and MSnSet.

Examples

```
quantFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "mzXML$")
identFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "dummyiTRAQ.mzid")
msexp <- readMSData(quantFile)
msexp <- addIdentificationData(msexp, identFile)
fData(msexp)$pepseq
length(msexp)

## using default fcol
msexp2 <- removeNoId(msexp)
length(msexp2)
fData(msexp2)$pepseq

## using keep
print(fvarLabels(msexp))
(k <- fData(msexp)$'ms-gf:evalue' > 75)
k[is.na(k)] <- FALSE
k
msexp3 <- removeNoId(msexp, keep = k)
length(msexp3)
fData(msexp3)$pepseq
```

---

removePeaks-methods *Removes low intensity peaks*

Description

This method sets low intensity peaks from individual spectra (Spectrum instances) or whole experiments (MSnExp instances) to 0. The intensity threshold is set with the t parameter. Default is the "min" character. The threshold is then set as the non-0 minimum intensity found in the spectrum. Any other numeric values is valid. All peaks with maximum intensity smaller or equal to t are set to 0.

If the spectrum is in profile mode, ranges of successive non-0 peaks <= t are set to 0. If the spectrum is centroided, then individual peaks <= t are set to 0. See the example below for an illustration.

Note that the number of peaks is not changed; the peaks below the threshold are set to 0 and the object is not cleanded out (see [clean](#)). An illustrative example is shown below.

Methods

signature(object = "MSnExp", t, verbose = "logical") Removes low intensity peaks of all spectra in MSnExp object. t sets the minimum peak intensity. Default is "min", i.e the smallest intensity in each spectrum. Other numeric values are valid. Displays a control bar if verbose set to TRUE (default). Returns a new MSnExp instance.

signature(object = "Spectrum", t) Removes low intensity peaks of Spectrum object. t sets the minimum peak intensity. Default is "min", i.e the smallest intensity in each spectrum. Other numeric values are valid. Returns a new Spectrum instance.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

[clean](#) and [trimMz](#) for other spectra processing methods.

Examples

```
int <- c(2,0,0,0,1,5,1,0,0,1,3,1,0,0,1,4,2,1)
sp1 <- new("Spectrum2",
           intensity=int,
           mz=1:length(int))
sp2 <- removePeaks(sp1) ## no peaks are removed here
                        ## as min intensity is 1 and
                        ## no peak has a max int <= 1
sp3 <- removePeaks(sp1,3)
intensity(sp1)
intensity(sp2)
intensity(sp3)
```

```
peaksCount(sp1) == peaksCount(sp2)
peaksCount(sp3) <= peaksCount(sp1)

data(itraqdata)
itraqdata2 <- removePeaks(itraqdata, t = 2.5e5)
table(unlist(intensity(itraqdata)) == 0)
table(unlist(intensity(itraqdata2)) == 0)
processingData(itraqdata2)

## difference between centroided and profile peaks

int <- c(104, 57, 32, 33, 118, 76, 38, 39, 52, 140, 52, 88, 394, 71,
         408, 94, 2032)
sp <- new("Spectrum2",
          intensity = int,
          centroided = FALSE,
          mz = seq_len(length(int)))

## unchanged, as ranges of peaks <= 500 considered
intensity(removePeaks(sp, 500))


centroided(sp) <- TRUE
## different!
intensity(removePeaks(sp, 500))
```

---

removeReporters-methods

Removes reporter ion tag peaks

Description

This methods sets all the reporter tag ion peaks from one MS2 spectrum or all the MS2 spectra of an experiment to 0. Reporter data is specified using an *"ReporterIons"* instance. The peaks are selected around the expected reporter ion m/z value +/- the reporter width. Optionally, the spectrum/spectra can be cleaned to remove successive 0 intensity data points (see the clean function for details).

Note that this method only works for MS2 spectra or experiments that contain MS2 spectra. It will fail for MS1 spectrum.

Methods

signature(object = "MSnExp", reporters = "ReporterIons", clean = "logical", verbose = "logical")
    The reporter ion peaks defined in the reporters instance of all the MS2 spectra of the
    "MSnExp" instance are set to 0 and, if clean is set to TRUE, cleaned. The default value of
    reporters is NULL, which leaves the spectra as unchanged. The verbose parameter (default
    is TRUE) defines whether a progress bar should be showed.

signature(object = "Spectrum", reporters = "ReporterIons", clean = "FALSE") The reporter ion peaks defined in the reporters instance of MS2 "Spectrum" instance are set to 0 and, if clean is set to TRUE, cleaned. The default value of reporters is NULL, which leaves the spectrum as unchanged.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

clean and removePeaks for other spectra processing methods.

Examples

```
sp1 <- itraqdata[[1]]
sp2 <- removeReporters(sp1,reporters=iTRAQ4)
sel <- mz(sp1) > 114 & mz(sp1) < 114.2
mz(sp1)[sel]
intensity(sp1)[sel]
plot(sp1,full=TRUE,reporters=iTRAQ4)
intensity(sp2)[sel]
plot(sp2,full=TRUE,reporters=iTRAQ4)
```

ReporterIons-class   *The "ReporterIons" Class*

Description

The ReporterIons class allows to define a set of isobaric reporter ions that are used for quantification in MSMS mode, e.g. iTRAQ (isobaric tag for relative and absolute quantitation) or TMT (tandem mass tags). ReporterIons instances can them be used when quantifying "MSnExp" data of plotting the reporters peaks based on in "Spectrum2" ojects.

Some reporter ions are provided with MSnbase an can be loaded with the data function. These reporter ions data sets are:

iTRAQ4: ReporterIon object for the iTRAQ 4-plex set. Load with data(iTRAQ4).

iTRAQ5: ReporterIon object for the iTRAQ 4-plex set plus the isobaric tag. Load with data(iTRAQ5).

TMT6: ReporterIon object for the TMT 6-plex set. Load with data(TMT6).

TMT7: ReporterIon object for the TMT 6-plex set plus the isobaric tag. Load with data(TMT6).

Objects from the Class

Objects can be created by calls of the form new("ReporterIons", ...).

Slots

name: Object of class "character" to identify the ReporterIons instance.

reporterNames: Object of class "character" naming each individual reporter of the ReporterIons instance. If not provided explicitly, they are names by concatenating the ReporterIons name and the respective MZ values.

description: Object of class "character" to describe the ReporterIons instance.

mz: Object of class "numeric" providing the MZ values of the reporter ions.

col: Object of class "character" providing colours to highlight the reporters on plots.

width: Object of class "numeric" indicating the width around the individual reporter ions MZ values were to search for peaks. This is dependent on the mass spectrometer's resolution and is used for peak picking when quantifying the reporters. See [quantify](#) for more details about quantification.

.__classVersion__: Object of class "Versions" indicating the version of the ReporterIons instance. Intended for developer use and debugging.

Extends

Class ["Versioned"](#), directly.

Methods

show(object) Displays object content as text.

object[] Subsets one or several reporter ions of the ReporterIons object and returns a new instance of the same class.

length(object) Returns the number of reporter ions in the instance.

mz(object, ...) Returns the expected mz values of reporter ions. Additional arguments are currently ignored.

reporterColours(object) **or reporterColors(object)** Returns the colours used to highlight the reporter ions.

reporterNames(object) Returns the name of the individual reporter ions. If not specified or is an incorrect number of names is provided at initialisation, the names are generated automatically by concatenating the instance name and the reporter's MZ values.

reporterNames(object) <- value Sets the reporter names to value, which must be a character of the same length as the number of reporter ions.

width(object) Returns the widths in which the reporter ion peaks are expected.

names(object) Returns the name of the ReporterIons object.

description(object) Returns the description of the ReporterIons object.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

Ross PL, Huang YN, Marchese JN, Williamson B, Parker K, Hattan S, Khainovski N, Pillai S, Dey S, Daniels S, Purkayastha S, Juhasz P, Martin S, Bartlet-Jones M, He F, Jacobson A, Pappin DJ. "Multiplexed protein quantitation in Saccharomyces cerevisiae using amine-reactive isobaric tagging reagents." *Mol Cell Proteomics*, 2004 Dec;3(12):1154-69. Epub 2004 Sep 22. PubMed PMID: 15385600.

Thompson A, Sch\"afer J, Kuhn K, Kienle S, Schwarz J, Schmidt G, Neumann T, Johnstone R, Mohammed AK, Hamon C. "Tandem mass tags: a novel quantification strategy for comparative analysis of complex protein mixtures by MS/MS." *Anal Chem.* 2003 Apr 15;75(8):1895-904. *Erratum* in: *Anal Chem.* 2006 Jun 15;78(12):4235. Mohammed, A Karim A [added] and *Anal Chem.* 2003 Sep 15;75(18):4942. Johnstone, R [added]. PubMed PMID: 12713048.

See Also

TMT6 or iTRAQ4 for readily available examples.

Examples

```
## Code used for the iTRAQ4 set
ri <- new("ReporterIons",
          description="4-plex iTRAQ",
          name="iTRAQ4",
          reporterNames=c("iTRAQ4.114","iTRAQ4.115",
                          "iTRAQ4.116","iTRAQ4.117"),
          mz=c(114.1,115.1,116.1,117.1),
          col=c("red","green","blue","yellow"),
          width=0.05)
ri
reporterNames(ri)
ri[1:2]
```

---

smooth-methods   *Smooths 'MSnExp' or 'Spectrum' instances*

Description

This method smooths individual spectra (Spectrum instances) or whole experiments (MSnExp instances). Currently, the Savitzky-Golay-Smoothing (method = "SavitzkyGolay") and the Moving-Average-Smoothing (method = "MovingAverage") are available, as implemented in the MALDIquant::smoothIntensity function. Additional methods might be added at a later stage.

Methods

signature(x = "MSnExp", method = "character", halfWindowSize = "integer", verbose = "logical", ...)
    Smooths all spectra in MSnExp. method could be "SavitzkyGolay" or "MovingAverage}. \code{halfWindowSize}
    controls the window size of the filter. The resulting window size is
    \code{2 * halfWindowSize + 1}. The best size differs depending on   the selected \code{method}.

should be lower than \emph{FWHM} of the peaks (full width   at half maximum; please find detail
The arguments \code{...} are passed to the internal functions.   Currently the only supported a
(default: 3) for \code{method="SavitzkyGolay to control the polynomial order of the
Savitzky-Golay Filter. This method displays a progress bar if verbose = TRUE. Returns an
MSnExp instance with smoothed spectra.

signature(x = "Spectrum", method = "character", halfWindowSize = "integer", ...)
Smooths the spectrum (Spectrum instance). This method is the same as above but returns a
smoothed Spectrum instead of an MSnExp object. It has no verbose argument. Please read
the details for the above MSnExp method.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

References

A. Savitzky and M. J. Golay. 1964. Smoothing and differentiation of data by simplified least squares
procedures. Analytical chemistry, 36(8), 1627-1639.

M. U. Bromba and H. Ziegler. 1981. Application hints for Savitzky-Golay digital smoothing filters.
Analytical Chemistry, 53(11), 1583-1586.

S. Gibb and K. Strimmer. 2012. MALDIquant: a versatile R package for the analysis of mass spec-
trometry data. Bioinformatics 28: 2270-2271. http://strimmerlab.org/software/maldiquant/

See Also

clean, pickPeaks, removePeaks and trimMz for other spectra processing methods.

Examples

```
sp1 <- new("Spectrum1",
           intensity = c(1:6, 5:1),
           mz = 1:11)
sp2 <- smooth(sp1, method = "MovingAverage", halfWindowSize = 2)
intensity(sp2)

data(itraqdata)
itraqdata2 <- smooth(itraqdata,
                     method = "MovingAverage",
                     halfWindowSize = 2)
processingData(itraqdata2)
```

Spectrum-class   *The "Spectrum" Class*

Description

Virtual container for spectrum data common to all different types of spectra. A Spectrum object
can not be directly instanciated. Use "Spectrum1" and "Spectrum2" instead.

Slots

msLevel: Object of class "integer" indicating the MS level: 1 for MS1 level Spectrum1 objects and 2 for MSMSM Spectrum2 objects. Levels > 2 have not been tested and will be handled as MS2 spectra.

peaksCount: Object of class "integer" indicating the number of MZ peaks.

rt: Object of class "numeric" indicating the retention time (in seconds) for the current ions.

tic: Object of class "numeric" indicating the total ion current.

acquisitionNum: Object of class "integer" corresponding to the acquisition number of the current spectrum.

scanIndex: Object of class "integer" indicating the scan index of the current spectrum.

mz: Object of class "numeric" of length equal to the peaks count (see peaksCount slot) indicating the MZ values that have been measured for the current ion.

intensity: Object of class "numeric" of same length as mz indicating the intensity at which each mz datum has been measured.

centroided: Object of class "logical" indicating if instance is centroided ('TRUE') of uncentroided ('FALSE').

fromFile: Object of class "integer" referencing the file the spectrum originates. The file names are stored in the processingData slot of the ["MSnExp"](#) or ["MSnSet"](#) instance that contains the current "Spectrum" instance.

.__classVersion__: Object of class "Versions" indicating the version of the Spectrum class. Intended for developer use and debugging.

Extends

Class ["Versioned"](#), directly.

Methods

acquisitionNum(object) Returns the acquisition number of the spectrum as an integer.

scanIndex(object) Returns the scan index of the spectrum as an integer.

centroided(object) Indicates whether spectrum is centroided ('TRUE') or uncentroided ('FALSE').

centroided(object) <- value Sets the 'centroided' status of the spectrum object.

fromFile(object) Returns the index of the raw data file from which the current instances originates as an integer.

intensity(object) Returns an object of class "numeric" containing the intensities of the spectrum.

msLevel(object) Returns an MS level of the spectrum as an integer.

mz(object, ...) Returns an object of class "numeric" containing the MZ value of the spectrum peaks. Additional arguments are currently ignored.

peaksCount(object) Returns the number of peaks (possibly of 0 intensity) as an integer.

rtime(object, ...) Returns the retention time for the spectrum as an integer. Additional arguments are currently ignored.

ionCount(object) Returns the total ion count for the spectrum as a numeric.

tic(object, ...) Returns the total ion current for the spectrum as a numeric. Additional arguments are currently ignored.

bin signature(object = "Spectrum"): Bins Spectrum. See [bin](#) documentation for more details and examples.

clean signature(object = "Spectrum"): Removes unused 0 intensity data points. See [clean](#) documentation for more details and examples.

compareSpectra signature(object1 = "Spectrum",   object2 = "Spectrum"): Compares spectra. See [compareSpectra](#) documentation for more details and examples.

pickPeaks signature(object = "Spectrum"): Performs the peak picking to generate a centroided spectrum. See [pickPeaks](#) documentation for more details and examples.

plot signature(x = "Spectrum", y = "missing"): Plots intensity against mz. See [plot.Spectrum](#) documentation for more details.

plot signature(x = "Spectrum", y = "Spectrum"): Plots two spectra above/below each other. See [plot.Spectrum.Spectrum](#) documentation for more details.

plot signature(x = "Spectrum", y = "character"): Plots an MS2 level spectrum and its highlight the fragmention peaks. See [plot.Spectrum.character](#) documentation for more details.

quantify signature(object = "Spectrum"): Quatifies defined peaks in the spectrum. See [quantify](#) documentation for more details.

removePeaks signature(object = "Spectrum"): Remove peaks lower that a threshold t. See [removePeaks](#) documentation for more details and examples.

smooth signature(x = "Spectrum"): Smooths spectrum. See [smooth](#) documentation for more details and examples.

show signature(object = "Spectrum"): Displays object content as text.

trimMz signature(object = "Spectrum"): Trims the MZ range of all the spectra of the MSnExp instance. See [trimMz](#) documentation for more details and examples.

isEmpty signature(x = "Spectrum"): Checks if the x is an empty Spectrum.

as signature(object = "Spectrum", "data.frame"): Coerces the Spectrum object to a two-column data.frame containing intensities and MZ values.

Note

This is a virtual class and can not be instanciated directly.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

Instaciable sub-classes ["Spectrum1"](#) and ["Spectrum2"](#) for MS1 and MS2 spectra.

Spectrum1-class *The "Spectrum1" Class for MS1 Spectra*

Description

Spectrum1 extends the "Spectrum" class and introduces an MS1 specific attribute in addition to the slots in "Spectrum". Spectrum1 instances are not created directly but are contained in the assayData slot of an "MSnExp".

Slots

polarity: Object of class "integer" indicating the polarity if the ion.

msLevel: Object of class "integer" indicating the MS level: always 1 in this case (inherited from "Spectrum").

peaksCount: Object of class "integer" indicating the number of MZ peaks (inherited from "Spectrum").

rt: Object of class "numeric" indicating the retention time (in seconds) for the current ions (inherited from "Spectrum").

acquisitionNum: Object of class "integer" corresponding to the acquisition number of the current spectrum (inherited from "Spectrum").

scanIndex: Object of class "integer" indicating the scan index of the current spectrum (inherited from "Spectrum").

mz: Object of class "numeric" of length equal to the peaks count (see peaksCount slot) indicating the MZ values that have been measured for the current ion (inherited from "Spectrum").

intensity: Object of class "numeric" of same length as mz indicating the intensity at which each mz datum has been measured "Spectrum").

.__classVersion__: Object of class "Versions" indicating the versions of the Spectrum and Spectrum1 classes of the instance. Intended for developer use and debugging.

Extends

Class "Spectrum", directly. Class "Versioned", by class "Spectrum", distance 2.

Methods

See "Spectrum" for additional accessors and methods to process Spectrum1 objects.

polarity(object) Returns the polarity of the spectrum as an integer.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

Virtual super-class "Spectrum", "Spectrum2" for MS2 spectra and "MSnExp" for a full experiment container.

Spectrum2-class   *The "Spectrum2" Class for MSMS Spectra*

Description

Spectrum2 extends the "[Spectrum](#)" class and introduces several MS2 specific attributes in addition to the slots in "[Spectrum](#)". Spectrum2 are not created directly but are contained in the assayData slot of an "[MSnExp](#)".

Slots

merged: Object of class "numeric" indicating of how many combination the current spectrum is the result of.

precScanNum: Object of class "integer" indicating the precursor MS scan index in the original input file. Accessed with the precScanNum or precAcquisitionNum methods.

precursorMz: Object of class "numeric" providing the precursor ion MZ value.

precursorIntensity: Object of class "numeric" providing the precursor ion intensity.

precursorCharge: Object of class "integer" indicating the precursor ion charge.

collisionEnergy: Object of class "numeric" indicating the collision energy used to fragment the parent ion.

msLevel: Object of class "integer" indicating the MS level: 2 in this case (inherited from "[Spectrum](#)").

peaksCount: Object of class "integer" indicating the number of MZ peaks (inherited from "[Spectrum](#)").

rt: Object of class "numeric" indicating the retention time (in seconds) for the current ions (inherited from "[Spectrum](#)").

acquisitionNum: Object of class "integer" corresponding to the acquisition number of the current spectrum (inherited from "[Spectrum](#)").

scanIndex: Object of class "integer" indicating the scan index of the current spectrum (inherited from "[Spectrum](#)").

mz: Object of class "numeric" of length equal to the peaks count (see peaksCount slot) indicating the MZ values that have been measured for the current ion (inherited from "[Spectrum](#)").

intensity: Object of class "numeric" of same length as mz indicating the intensity at which each mz datum has been measured "[Spectrum](#)").

.__classVersion__: Object of class "Versions" indicating the versions of the Spectrum and Spectrum2 classes of the instance. Intended for developer use and debugging.

Extends

Class "[Spectrum](#)", directly. Class "[Versioned](#)", by class "Spectrum", distance 2.

Methods

See *"*Spectrum*"* for additional accessors and methods for Spectrum2 objects.

precursorMz(object) Returns the precursor MZ value as a numeric.

precursorMz(object) Returns the precursor scan number in the original data file as an integer.

precursorIntensity(object) Returns the precursor intensity as a numeric.

precursorCharge(object) Returns the precursor intensity as a integer.

collisionEnergy(object) Returns the collision energy as an numeric.

removeReporters(object, ...) Removes all reporter ion peaks. See removeReporters documentation for more details and examples.

precAcquisitionNum: Returns the precursor's acquisition number.

precScanNum: See precAcquisitionNum.

calculateFragments signature(sequence = "character", object = "Spectrum2"): Calculates and matches the theoretical fragments of a peptide sequence with the ones observed in a spectrum. See calculateFragments documentation for more details and examples.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

Virtual super-class *"*Spectrum*"*, *"*Spectrum1*"* for MS1 spectra and *"*MSnExp*"* for a full experiment container.

TMT6 *TMT 6-plex set*

Description

This instance of class *"*ReporterIons*"* corresponds to the TMT 6-plex set, i.e the 126, 127, 128, 129, 130 and 131 isobaric tags. In the TMT7 data set, an unfragmented tag, i.e reporter and attached isobaric tag, is also included at MZ 229. The TMT10 instance corresponds to the 10-plex version.

These objects are used to plot the reporter ions of interest in an MSMS spectra (see *"*Spectrum2*"*) as well as for quantification (see quantify).

Usage

```
TMT6
TMT7
TMT10
```

References

Thompson A, Sch\"afer J, Kuhn K, Kienle S, Schwarz J, Schmidt G, Neumann T, Johnstone R, Mohammed AK, Hamon C. "Tandem mass tags: a novel quantification strategy for comparative analysis of complex protein mixtures by MS/MS." *Anal Chem.* 2003 Apr 15;75(8):1895-904. *Erratum* in: *Anal Chem.* 2006 Jun 15;78(12):4235. Mohammed, A Karim A [added] and *Anal Chem.* 2003 Sep 15;75(18):4942. Johnstone, R [added]. PubMed PMID: 12713048.

See Also

iTRAQ4.

Examples

```
TMT6
TMT6[1:2]

TMT10

newReporter <- new("ReporterIons",
                    description="an example",
                    name="my reporter ions",
                    reporterNames=c("myrep1","myrep2"),
                    mz=c(121,122),
                    col=c("red","blue"),
                    width=0.05)
newReporter
```

---

trimMz-methods   *Trims 'MSnExp' or 'Spectrum' instances*

Description

This method selects a range of MZ values in a single spectrum (Spectrum instances) or all the spectra of an experiment (MSnExp instances). The regions to trim are defined by the range of mzlim argument, such that MZ values < min(mzlim) and MZ values > max(mzlim) are trimmed away.

Methods

signature(object = "MSnExp", mzlim = "numeric") Trims all spectra in MSnExp object according to mzlim. Returns a cleaned MSnExp instance.

signature(object = "Spectrum", mzlim = "numeric") Trims the Spectrum object and retruns a new trimmed object.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

removePeaks and clean for other spectra processing methods.

Examples

```
mz <- 1:100
sp1 <- new("Spectrum2",
           mz=mz,
           intensity=abs(rnorm(length(mz))))
sp2 <- trimMz(sp1,c(25,75))
range(mz(sp1))
range(mz(sp2))

data(itraqdata)
itraqdata2 <- trimMz(itraqdata,c(113,117))
range(mz(itraqdata))
range(mz(itraqdata2))
processingData(itraqdata2)
```

---

writeMgfData-methods   *Write an experiment or spectrum to an mgf file*

Description

Methods writeMgfData write individual "Spectrum" instances of whole "MSnExp" experiments to a file in Mascot Generic Format (mgf) (see http://www.matrixscience.com/help/data_file_help.html for more details). Function readMgfData read spectra from and mgf file and creates an "MSnExp" object.

Arguments

object   An instance of class "Spectrum" or "MSnExp".

con   A valid connection or a character string with the name of the file to save the object. In case of the latter, a file connection is created. If not specified, 'spectrum.mgf' or 'experiment.mgf' are used depending on the class of object. Note that existing files are overwritted.

COM   Optional character vector with the value for the 'COM' field.

TITLE   Optional character vector with the value for the spectrum 'TITLE' field. Not applicable for experiments.

Details

Note that when reading an mgf file, the original order of the spectra is lost. Thus, if the data was originally written to mgf from an MSnExp object using writeMgfData, although the feature names will be identical, the spectra are not as a result of the reordering. See example below.

Methods

signature(object = ″MSnExp″) Writes the full exeriment to an mgf file.

signature(object = ″Spectrum″) Writes an individual spectrum to an mgf file.

See Also

[readMgfData](#) function to read data from and mgf file.

Examples

```
## Not run:
  data(itraqdata)
  writeMgfData(itraqdata,file="itraqdata.mgf",COM="MSnbase itraqdata")
  itraqdata2 <- readMgfData("itraqdata.mgf")
  ## note that the order of the spectra
  ## and precision of some values (precursorMz for instance)
  ## are altered
  match(signif(precursorMz(itraqdata2),4),signif(precursorMz(itraqdata),4))
  ## [1]  1 10 11 12 13 14 15 16 17 18 ...
  ## ... but all the precursors are there
  all.equal(sort(precursorMz(itraqdata2)),sort(precursorMz(itraqdata)),
            check.attributes=FALSE,
            tolerance=10e-5)
  ## is TRUE
  all.equal(as.data.frame(itraqdata2[[1]]),as.data.frame(itraqdata[[1]]))
  ## is TRUE
  all.equal(as.data.frame(itraqdata2[[3]]),as.data.frame(itraqdata[[11]]))
  ## is TRUE
  ## But, beware that
  all(featureNames(itraqdata2)==featureNames(itraqdata))
  ## is TRUE too!

## End(Not run)
```

---

writeMzTabData   *Writes an 'MSnSet' to an mzTab file*

Description

This function generates an mzTab file based on the data available in teh x MSnSet instance and additional information passed by the user. It make used of the respective section generators to create appropriate metadata, peptide and protein sections. If peptide and protein sections need to be generated, one has to first create the mzTab file with (metadata, optional, but recommended and) protein data and then append the peptide data, as per mzTab specification (http://code.google.com/p/mztab/). See the example section.

Usage

```
writeMzTabData(x, what = c("PEP", "PRT"), append = FALSE, MTD = TRUE,
    file, ...)
```

Arguments

x	An instance of class MSnSet.
what	One of "PEP" or "PRT" defining whether peptide or protein data is to be saved.
append	Logical. Should the data be appended to file. Default is FALSE.
MTD	Logical. Should the metadata section be generated. Default is TRUE.
file	A character naming the file to print to.
...	Additional parameters passed to the respective section generators: makeMTD for metadata, makePEP for peptides and makePRT for proteins.

Value

None (invisible NULL).

Author(s)

Laurent Gatto

References

The mzTab specification document and example files: <http://code.google.com/p/mztab/>.

See Also

Functions to generate metadata (makeMTD), peptide data (makePEP) and proteins (makePRT). readMzTabData to create "MSnSet" instances from an mzTab file.

Examples

```
mzTabFile <- tempfile()
data(msnset)
pep <- msnset
prot <- combineFeatures(pep, groupBy = fData(pep)$ProteinAccession)
fvarLabels(pep)
## First write metadata and protein data
writeMzTabData(prot, what = "PRT", file = mzTabFile,
               append = FALSE, MTD = TRUE,
               protAccession = fData(prot)$ProteinAccession,
               protDescription = fData(prot)$ProteinDescription,
               protAbundance = exprs(prot))
## append peptide data, without metadata section
writeMzTabData(pep, what = "PEP", file = mzTabFile,
               append = TRUE, MTD = FALSE,
               sequence = fData(pep)$PeptideSequence,
               charge = fData(pep)$charge,
```

```
                 retentionTime = fData(pep)$retention.time,
                 pepAbundance = exprs(pep))
```

xic-methods   *Extracted ion chromatograms*

Description

These methods produce an extracted ion chromatogram given mass-spectrometry data and an ion to be extracted. In addition to the object (see 'Methods' section below), additional arguments are

mz A numeric specifying the ion mass to be extracted.

width The M/Z extraction with. Default is 0.5.

rtlim The retention time limit to be displayed. When missing (default), the complete range of the matching extracted ions is plotted.

npeaks The number of peaks to be annotated. Default is 3.

charge The charge of the ion to be extracted. This value is optional; when provided, the mass of the ion (mz above) will first be divided by its charge before extraction.

clean A logical, defining of the XIC data should be cleaned before plotting. Default if TRUE. See [clean](#) for details.

legend A logical defining if the figure should be annotated.

plot A logical defining if the plot should be rendered.

points A logical specifying if points should be added to for each individual MS spectrum. Annotated peaks are coloured in orange and MS2 spectra with precursor M/Z matching the extracted ion are highlighted in red.

hd The header data.frame corresponding to the object input. hd is generated automatically and can generally be omitted.

... Additional arguments passed to the plot function.

Note: the methods are currently not vectorised.

xcms::plotEIC provides a similar functionality.

Value

The methods invisibly return the data.frame with the the XIC intensities (column int), retention times (column rt) and extracted M/Z values (column mz) used to generate the figure.

Methods

signature(object = "character") Plots the XIC for the mass-spectrometry data stored in the object file. The file format must be support by mzR. See mzR::openMSfile for details.

signature(object = "mzRramp") Plots the XIC for the mzRramp instance. See the mzR package for details.

Examples

```
## Not run:
    library("RforProteomics")
    (f <- getPXD000001mzXML())
    ms <- openMSfile(f)
    x <- xic(ms, mz = 636.925, width = 0.01)
    head(x)
    xic(ms, mz = 636.925, width = 0.01,
        npeaks = 1,
        rtlim = c(2000, 2300))

## End(Not run)
```

Index