Package 'MSnbase'

November 12, 2026

Title Base Functions and Classes for Mass Spectrometry and Proteomics

Version 2.38.1

Description MSnbase provides infrastructure for manipulation, processing and visualisation of mass spectrometry and proteomics data, ranging from raw to quantitative and annotated data.

Author Laurent Gatto, Johannes Rainer and Sebastian Gibb with contributions from Guangchuang Yu, Samuel Wieczorek, Vasile-Cosmin Lazar, Vladislav Petyuk, Thomas Naake, Richie Cotton, Arne Smits, Martina Fisher, Ludger Goeminne, Adriaan Sticker, Lieven Clement and Pascal Maas.

Maintainer Laurent Gatto <laurent.gatto@uclouvain.be>

Depends R (>= 3.5), methods, BiocGenerics (>= 0.7.1), Biobase (>= 2.15.2), mzR (>= 2.29.3), S4Vectors, ProtGenerics (>= 1.29.1)

Imports MsCoreUtils, PSMatch (>= 1.15.3), PTMods (>= 0.99.5), BiocParallel, IRanges (>= 2.13.28), plyr, vsn, grid, stats4, affy, impute, pcaMethods, MALDIquant (>= 1.16), mzID (>= 1.5.2), digest, lattice, ggplot2, scales, MASS, Rcpp

Suggests testthat, gridExtra, microbenchmark, zoo, knitr (>= 1.1.0), Rdisop, pRoloc, pRolocdata (>= 1.43.3), magick, MsDataHub, msdata, roxygen2, rgl, rpx, AnnotationHub, BiocStyle (>= 2.5.19), rmarkdown, imputeLCMD, norm, gplots, XML, shiny, magrittr, SummarizedExperiment, Spectra

LinkingTo Rcpp

License Artistic-2.0

LazyData yes

VignetteBuilder knitr

Encoding UTF-8

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

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

biocViews ImmunoOncology, Infrastructure, Proteomics, MassSpectrometry, QualityControl, DataImport

Roxygen list(markdown=TRUE)

Collate 'AllClassUnions.R' 'AllGenerics.R' 'DataClasses.R' 'MzTab.R' 'NTR.R' 'RcppExports.R' 'TMT10.R' 'TMT11.R' 'TMT16.R' 'TMT6.R' 'TMT7.R' 'averageMSnSet.R' 'cache.R' 'coerce.R' 'combineFeatures.R' 'compfnames.R' 'environment.R' 'fData-utils.R' 'fdata-selection.R' 'foi.R' 'functions-Chromatogram.R' 'functions-MChromatograms.R' 'functions-MIAPE.R' 'functions-MSnExp.R' 'functions-MSnProcess.R' 'functions-MSnSet.R' 'functions-MSpectra.R' 'functions-OnDiskMSnExp.R' 'functions-ReporterIons.R' 'functions-Spectrum.R' 'functions-Spectrum1.R' 'functions-Spectrum2.R' 'functions-addIdentificationData.R' 'functions-mzR.R' 'functions-plotting.R' 'hmap.R' 'iPQF.R' 'iTRAQ4.R' 'iTRAQ5.R' 'iTRAQ8.R' 'iTRAQ9.R' 'imputation.R' 'map.R' 'matching.R' 'methods-Chromatogram.R' 'methods-MChromatograms.R' 'methods-MIAPE.R' 'methods-MSnExp.R' 'methods-MSnProcess.R' 'methods-MSnSet.R' 'methods-MSnSetList.R' 'methods-MSpectra.R' 'methods-OnDiskMSnExp.R' 'methods-ReporterIons.R' 'methods-Spectrum.R' 'methods-Spectrum1.R' 'methods-Spectrum2.R' 'methods-all.equal.R' 'methods-filters.R' 'methods-mzR.R' 'methods-other.R' 'methods-pSet.R' 'methods-updateObjectTo.R' 'methods-write.R' 'missing-data.R' 'nadata.R' 'options.R' 'plotting-MSnExp.R' 'plotting-MSnSet.R' 'plotting-Spectrum.R' 'plotting-Spectrum1.R' 'plotting-Spectrum2.R' 'plotting-dataframe.R' 'quantitation-MS2-isobaric.R' 'quantitation-MS2-labelfree.R' 'readChromData.R' 'readMSData.R' 'readMSData2.R' 'readMSnSet.R' 'readMzXMLData.R' 'readWriteMgfData.R' 'readWriteMzTab.R' 'utils.R' 'writeMSData.R' 'zzz.R'

Config/roxygen2/version 8.0.0

git_url https://git.bioconductor.org/packages/MSnbase

git_branch RELEASE_3_23

git_last_commit c460ef9

git_last_commit_date 2026-08-07

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

addIdentificationData-methods 5
aggvar 7
as 8

averageMSnSet . . . 9
bin-methods . . . 11
calculateFragments-methods . . . 12
Chromatogram . . . 14
chromatogram,MSnExp-method . . . 22
clean-methods . . . 24
combineFeatures . . . 26
combineSpectra,MSnExp-method . . . 29
combineSpectraMovingWindow . . . 32
commonFeatureNames . . . 34
compareMSnSets . . . 35
compareSpectra-methods . . . 36
consensusSpectrum . . . 37
Deprecated . . . 39
estimateMzResolution,MSnExp-method . . . 39
estimateMzScattering . . . 41
estimateNoise-methods . . . 42
expandFeatureVars . . . 43
extractPrecSpectra-methods . . . 44
extractSpectraData . . . 45
factorsAsStrings . . . 46
FeatComp-class . . . 47
featureCV . . . 48
FeaturesOfInterest-class . . . 49
fillUp . . . 52
filterIdentificationDataFrame . . . 53
formatRt . . . 54
getVariableName . . . 54
grepEcols . . . 55
hasSpectra . . . 56
imageNA2 . . . 57
impute,MSnSet-method . . . 58
iPQF . . . 59
isCentroidedFromFile . . . 61
iTRAQ4 . . . 62
itraqdata . . . 63
listOf . . . 64
makeCamelCase . . . 64
makeNaData . . . 65
MChromatograms . . . 67
meanMzInts . . . 75
MIAPE-class . . . 78
missing-data . . . 80
MSmap-class . . . 82
MSnbaseOptions . . . 84
MSnExp-class . . . 85
MSnProcess-class . . . 89
MSnSet-class . . . 90

MSnSetList-class . . . 96
MSpectra . . . 99
MzTab-class . . . 105
naplot . . . 107
nFeatures . . . 108
normalise-methods . . . 109
normToReference . . . 110
npcv . . . 112
nQuants . . . 113
OnDiskMSnExp-class . . . 114
pickPeaks-methods . . . 121
plot-methods . . . 123
plot.Spectrum.Spectrum-methods . . . 125
plot2d-methods . . . 127
plotDensity-methods . . . 128
plotMzDelta-methods . . . 129
plotNA-methods . . . 130
precSelection . . . 131
ProcessingStep-class . . . 132
pSet-class . . . 133
purityCorrect-methods . . . 137
quantify-methods . . . 140
readMgfData . . . 143
readMSData . . . 144
readMSnSet . . . 146
readMzIdData . . . 148
readMzTabData . . . 149
readMzTabData_v0.9 . . . 150
readSRMData . . . 151
reduce,data.frame-method . . . 153
removeNoId-methods . . . 154
removePeaks-methods . . . 155
removeReporters-methods . . . 156
ReporterIons-class . . . 157
selectFeatureData . . . 159
smooth-methods . . . 161
Spectrum-class . . . 162
Spectrum1-class . . . 165
Spectrum2-class . . . 165
TMT6 . . . 167
trimMz-methods . . . 168
updateObject-methods . . . 169
writeMgfData-methods . . . 169
writeMSData,MSnExp,character-method . . . 170
writeMzTabData . . . 172

Index **173**

addIdentificationData-methods
Adds Identification Data

Description

These methods add identification data to a raw MS experiment (an "MSnExp" object) or to quantitative data (an "MSnSet" object). The identification data needs to be available as a mzIdentML file (and passed as filenames, or directly as identification object) or, alternatively, can be passed as an arbitrary data.frame. See details in the *Methods* section.

Details

The featureData slots in a "MSnExp" or a "MSnSet" instance provides only one row per MS2 spectrum but the identification is not always bijective. Prior to addition, the identification data is filtered as documented in the filterIdentificationDataFrame function: (1) only PSMs matching the regular (non-decoy) database are retained; (2) PSMs of rank greater than 1 are discarded; and (3) only proteotypic peptides are kept.

If after filtering, more then one PSM per spectrum are still present, these are combined (reduced, see reduce,data.frame-method) into a single row and separated by a semi-colon. This has as side-effect that feature variables that are being reduced are converted to characters. See the reduce manual page for examples.

See also the section about identification data in the *MSnbase-demo* vignette for details and additional examples.

After addition of the identification data, new feature variables are created. 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. 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). All these values are re-calculated after filtering and reduction.

Methods

signature(object = "MSnExp", id = "character", ... 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. See below for additional arguments.

signature(object = "MSnExp", id = "mzID", ...) Same as above but id is a mzID object generated by mzID::mzID. See below for additional arguments.

signature(object = "MSnExp", id = "mzIDCollection", ...) Same as above but id is a mzIDCollection object. See below for additional arguments.

signature(object = "MSnExp", id = "mzRident", ... Same as above but id is a mzRident object generated by mzR::openIdfile. See below for additional arguments.

signature(object = "MSnExp", id = "data.frame", ... Same as above but id could be a data.frame. See below for additional arguments.

`signature(object = "MSnSet", id = "character", ...)` Adds the identification 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. See below for additional arguments.

`signature(object = "MSnSet", id = "mzID", ...)` Same as above but `id` is a `mzID` object. See below for additional arguments.

`signature(object = "MSnSet", id = "mzIDCollection", ...)` Same as above but `id` is a `mzIDCollection` object. See below for additional arguments.

`signature(object = "MSnSet", id = "data.frame", ...)` Same as above but `id` is a `data.frame`. See below for additional arguments.

The methods above take the following additional argument. These need to be set when adding identification data as a `data.frame`. In all other cases, the defaults are set automatically.

fcol The matching between the features (raw spectra or quantiative features) and identification results is done by matching columns in the featue data (the `featureData` slot) and the identification data. These values are the spectrum file index and the acquisition number, passed as a `character` of length 2. The default values for these variables in the `object`'s feature data are `"spectrum.file"` and `"acquisition.num"`. Values need to be provided when `id` is a `data.frame`.

icol The default values for the spectrum file and acquisition numbers in the identification data (the `id` argument) are `"spectrumFile"` and `"acquisitionNum"`. Values need to be provided when `id` is a `data.frame`.

acc The protein (group) accession number or identifier. Defaults are `"DatabaseAccess"` when passing filenames or `mzRident` objects and `"accession"` when passing `mzID` or `mzIDCollection` objects. A value needs to be provided when `id` is a `data.frame`.

desc The protein (group) description. Defaults are `"DatabaseDescription"` when passing filenames or `mzRident` objects and `"description"` when passing `mzID` or `mzIDCollection` objects. A value needs to be provided when `id` is a `data.frame`.

pepseq The peptide sequence variable name. Defaults are `"sequence"` when passing filenames or `mzRident` objects and `"pepseq"` when passing `mzID` or `mzIDCollection` objects. A value needs to be provided when `id` is a `data.frame`.

key The key to be used when the identification data need to be reduced (see details section). Defaults are `"spectrumID"` when passing filenames or `mzRident` objects and `"spectrumid"` when passing `mzID` or `mzIDCollection` objects. A value needs to be provided when `id` is a `data.frame`.

decoy The feature variable used to define whether the PSM was matched in the decoy of regular fasta database for PSM filtering. Defaults are `"isDecoy"` when passing filenames or `mzRident` objects and `"isdecoy"` when passing `mzID` or `mzIDCollection` objects. A value needs to be provided when `id` is a `data.frame`. See `filterIdentificationDataFrame` for details.

rank The feature variable used to defined the rank of the PSM for filtering. Defaults is `"rank"`. A value needs to be provided when `id` is a `data.frame`. See `filterIdentificationDataFrame` for details.

accession The feature variable used to defined the protein (groupo) accession or identifier for PSM filterin. Defaults is to use the same value as `acc` . A value needs to be provided when `id` is a `data.frame`. See `filterIdentificationDataFrame` for details.

verbose A `logical` defining whether to print out messages or not. Default is to use the session-wide open from `isMSnbaseVerbose`.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de> and Laurent Gatto

See Also

`filterIdentificationDataFrame` for the function that filters identification data, `readMzIdData` to read the identification data as a unfiltered `data.frame` and `reduce,data.frame-method` to reduce it to a `data.frame` that contains only unique PSMs per row.

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
fData(msexp)

idSummary(msexp)
```

---

aggvar *Identify aggregation outliers*

Description

This function evaluates the variability within all protein group of an `MSnSet`. If a protein group is composed only of a single feature, `NA` is returned.

Usage

```
aggvar(object, groupBy, fun)
```

Arguments

`object`	An object of class `MSnSet`.
`groupBy`	A `character` containing the protein grouping feature variable name.
`fun`	A function the summarise the distance between features within protein groups, typically `max` or `mean.median`.

Details

This function can be used to identify protein groups with incoherent feature (petides or PSMs) expression patterns. Using `max` as a function, one can identify protein groups with single extreme outliers, such as, for example, a mis-identified peptide that was erroneously assigned to that protein group. Using `mean` identifies more systematic inconsistencies where, for example, the subsets of peptide (or PSM) feautres correspond to proteins with different expression patterns.

Value

A `matrix` providing the number of features per protein group (`nb_feats` column) and the aggregation summarising distance (`agg_dist` column).

Author(s)

Laurent Gatto

See Also

`combineFeatures` to combine PSMs quantitation into peptides and/or into proteins.

Examples

```
library("pRolocdata")
data(hyperLOPIT2015ms3r1psm)
groupBy <- "Protein.Group.Accessions"
res1 <- aggvar(hyperLOPIT2015ms3r1psm, groupBy, fun = max)
res2 <- aggvar(hyperLOPIT2015ms3r1psm, groupBy, fun = mean)
par(mfrow = c(1, 3))
plot(res1, log = "y", main = "Single outliers (max)")
plot(res2, log = "y", main = "Overall inconsistency (mean)")
plot(res1[, "agg_dist"], res2[, "agg_dist"],
     xlab = "max", ylab = "mean")
```

---

as *Coerce identification data to a* `data.frame`

Description

A function to convert the identification data contained in an `mzRident` object to a `data.frame`. Each row represents a scan, which can however be repeated several times if the PSM matches multiple proteins and/or contains two or more modifications. To reduce the `data.frame` so that rows/scans are unique and use semicolon-separated values to combine information pertaining a scan, use `reduce`.

Arguments

`from`	An object of class `mzRident` defined in the `mzR` package.

Details

See also the *Tandem MS identification data* section in the *MSnbase-demo* vignette.

Value

A `data.frame`

Author(s)

Laurent Gatto

Examples

```
## find path to a mzIdentML file
identFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "dummyiTRAQ.mzid")
library("mzR")
x <- openIDfile(identFile)
x
as(x, "data.frame")
```

---

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 the mean after removing missing values, as computed by `function(x) mean(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 `NA`s). 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", msLevel. = "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. `msLevel.` defines the level of the spectrum, and if `msLevel(object) != msLevel.`, cleaning is ignored. Only relevant when called from `OnDiskMSnExp` and is only relevant for developers.

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", "z", "seq").

`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 `0.1`) and decide whether this tolerance should be applied relative to the target m/z (`relative = TRUE`) or absolute (default `relative = FALSE`) to match the theoretical fragment MZ with the MZ of the spectrum. When (`relative = TRUE`) the mass tolerance window is set to `target mz +/- (target mz * tolerance)` and `target mz +/- tolerance` otherwise. 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.names = TRUE, pattern = "mzXML$")

## create basic MSnExp
msexp <- readMSData(file, centroided = FALSE)

## centroid them
msexp <- pickPeaks(msexp)

## calculate fragments for ACE with default modification
calculateFragments("ACE")

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

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

## neutral loss
PSMatch::defaultNeutralLoss()

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

## real example
calculateFragments("PQR")
calculateFragments("PQR",
                   neutralLoss=PSMatch::defaultNeutralLoss(disableWaterLoss="Cterm"))
calculateFragments("PQR",
```

```
                    neutralLoss=PSMatch::defaultNeutralLoss(disableAmmoniaLoss="Q"))

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

Chromatogram *Representation of chromatographic MS data*

Description

The Chromatogram class is designed to store chromatographic MS data, i.e. pairs of retention time and intensity values. Instances of the class can be created with the Chromatogram constructor function but in most cases the dedicated methods for OnDiskMSnExp and MSnExp objects extracting chromatograms should be used instead (i.e. the chromatogram() method).

Usage

```
Chromatogram(
  rtime = numeric(),
  intensity = numeric(),
  mz = c(NA_real_, NA_real_),
  filterMz = c(NA_real_, NA_real_),
  precursorMz = c(NA_real_, NA_real_),
  productMz = c(NA_real_, NA_real_),
  fromFile = integer(),
  aggregationFun = character(),
  msLevel = 1L
)

aggregationFun(object)

## S4 method for signature 'Chromatogram'
show(object)

## S4 method for signature 'Chromatogram'
rtime(object)

## S4 method for signature 'Chromatogram'
intensity(object)

## S4 method for signature 'Chromatogram'
mz(object, filter = FALSE)

## S4 method for signature 'Chromatogram'
precursorMz(object)

## S4 method for signature 'Chromatogram'
```

```
fromFile(object)

## S4 method for signature 'Chromatogram'
length(x)

## S4 method for signature 'Chromatogram'
as.data.frame(x)

## S4 method for signature 'Chromatogram'
filterRt(object, rt)

## S4 method for signature 'Chromatogram'
clean(object, all = FALSE, na.rm = FALSE)

## S4 method for signature 'Chromatogram,ANY'
plot(
  x,
  col = "#00000060",
  lty = 1,
  type = "l",
  xlab = "retention time",
  ylab = "intensity",
  main = NULL,
  ...
)

## S4 method for signature 'Chromatogram'
msLevel(object)

## S4 method for signature 'Chromatogram'
isEmpty(x)

## S4 method for signature 'Chromatogram'
productMz(object)

## S4 method for signature 'Chromatogram'
bin(
  x,
  binSize = 0.5,
  breaks = seq(floor(min(rtime(x))), ceiling(max(rtime(x))), by = binSize),
  fun = max
)

## S4 method for signature 'Chromatogram'
normalize(object, method = c("max", "sum"))

## S4 method for signature 'Chromatogram'
filterIntensity(object, intensity = 0, ...)
```

```
## S4 method for signature 'Chromatogram,Chromatogram'
alignRt(x, y, method = c("closest", "approx"), ...)

## S4 method for signature 'Chromatogram,Chromatogram'
compareChromatograms(
  x,
  y,
  ALIGNFUN = alignRt,
  ALIGNFUNARGS = list(),
  FUN = cor,
  FUNARGS = list(use = "pairwise.complete.obs"),
  ...
)

## S4 method for signature 'Chromatogram'
transformIntensity(object, FUN = identity)
```

Arguments

rtime	for Chromatogram: numeric with the retention times (length has to be equal to the length of intensity).
intensity	for Chromatogram: numeric with the intensity values (length has to be equal to the length of rtime). For filterIntensity: numeric(1) or function to use to filter intensities. See description for details.
mz	for Chromatogram: numeric(2) representing the mz value range (min, max) on which the chromatogram was created. This is supposed to contain the *real* range of mz values in contrast to filterMz. If not applicable use mzrange = c(0, 0).
filterMz	for Chromatogram: numeric(2) representing the mz value range (min, max) that was used to filter the original object on m/z dimension. If not applicable use filterMz = c(0, 0).
precursorMz	for Chromatogram: numeric(2) for SRM/MRM transitions. Represents the mz of the precursor ion. See details for more information.
productMz	for Chromatogram: numeric(2) for SRM/MRM transitions. Represents the mz of the product. See details for more information.
fromFile	for Chromatogram: integer(1) the index of the file within the OnDiskMSnExp or MSnExp from which the chromatogram was extracted.
aggregationFun	for Chromatogram: character string specifying the function that was used to aggregate intensity values for the same retention time across the mz range. Supported are "sum" (total ion chromatogram), "max" (base peak chromatogram), "min" and "mean".
msLevel	for Chromatogram: integer(1) with the MS level from which the chromatogram was extracted.
object	Chromatogram object.
filter	for mz: logical(1) defining whether the m/z range to filter the originating object (e.g. MSnExp object) should be returned or the m/z range of the actual data. Defaults to filter = FALSE.

x	Chromatogram object.
rt	for filterRt: numeric(2) defining the lower and upper retention time to which the Chromatogram should be subsetted.
all	for clean: logical(1) whether all 0 intensities should be removed. Defaults to all = FALSE. See clean() for details.
na.rm	for clean: if all NA intensities should be removed before cleaning the Chromatogram. Defaults to clean = FALSE.
col	for plot: the color to be used for plotting.
lty	for plot: the line type. See help page of plot in the graphics package for details.
type	for plot: the type of plot. See help page of plot in the graphics package for details.
xlab	for plot: the x-axis label.
ylab	for plot: the y-axis label.
main	for plot: the plot title. If not provided the mz range will be used as plot title.
...	for plot: additional arguments to be passed to the base plot function. For filterIntensity: additional parameters passed along to the function provided with intensity. For compareChromatograms: ignored
binSize	for bin: numeric(1) with the size of the bins (in seconds). Defaults to binSize = 0.5.
breaks	for bin: numeric defining the bins. Usually not required as the function calculates the bins automatically based on binSize.
fun	for bin: function to be used to aggregate the intensity values falling within each bin. Defaults to fun = max.
method	character(1). For normalise: defining whether each chromatogram should be normalized to its maximum signal (method = "max") or total signal (method = "sum"). For alignRt: aligning approach that should be used (see description). Defaults to method = "closest".
y	for alignRt: Chromatogram against which x should be aligned against.
ALIGNFUN	for compareChromatograms: function to align chromatogram x against chromatogram y. Defaults to alignRt.
ALIGNFUNARGS	list of parameters to be passed to ALIGNFUN.
FUN	for compareChromatograms: function to calculate a similarity score on the intensity values of the compared and aligned chromatograms. Defaults to FUN = cor. For transformIntensity: function to transform chromatograms' intensity values. Defaults to FUN = identity.
FUNARGS	for compareChromatograms: list with additional parameters for FUN. Defaults to FUNARGS = list(use = "pairwise.complete.obs").

Details

The mz, filterMz, precursorMz and productMz are stored as a numeric(2) representing a range even if the chromatogram was generated for only a single ion (i.e. a single mz value). Using ranges for mz values allow this class to be used also for e.g. total ion chromatograms or base peak chromatograms.

The slots `precursorMz` and `productMz` allow to represent SRM (single reaction monitoring) and MRM (multiple SRM) chromatograms. As example, a `Chromatogram` for a SRM transition 273 -> 153 will have a `@precursorMz = c(273, 273)` and a `@productMz = c(153, 153)`.

Object creation

`Chromatogram` objects can be extracted from an `MSnExp` or `OnDiskMSnExp` object with the `chromatogram()` function.

Alternatively, the constructor function `Chromatogram` can be used, which takes arguments `rtime`, `intensity`, `mz`, `filterMz`, `precursorMz`, `productMz`, `fromFile`, `aggregationFun` and `msLevel`.

Data access and coercion

- `aggregationFun`: gets the aggregation function used to create the `Chromatogram`.
- `as.data.frame`: returns a `data.frame` with columns `"rtime"` and `"intensity"`.
- `fromFile`: returns an `integer(1)` with the index of the originating file.
- `intensity`: returns the intensities from the `Chromatogram`.
- `isEmpty`: returns `TRUE` if the chromatogram is empty or has only `NA` intensities.
- `length`: returns the length (i.e. number of data points) of the `Chromatogram`.
- `msLevel`: returns an `integer(1)` with the MS level of the chromatogram.
- `mz`: get the m/z (range) from the `Chromatogram`. The function returns a `numeric(2)` with the lower and upper boundaries. Parameter `filter` allows to specify whether the m/z range used to filter the originating object should be returned or the m/z range of the actual data.
- `precursorMz`: get the m/z of the precursor ion. The function returns a `numeric(2)` with the lower and upper boundary.
- `productMz`: get the m/z of the producto chromatogram/ion. The function returns a `numeric(2)` with the lower and upper m/z value.
- `rtime`: returns the retention times from the `Chromatogram`.

Data subsetting and filtering

- `filterRt`: filter/subset the `Chromatogram` to the specified retention time range (defined with parameter `rt`).
- `filterIntensity`: filter a `Chromatogram()` object removing data points with intensities below a user provided threshold. If `intensity` is a `numeric` value, the returned chromatogram will only contain data points with intensities > `intensity`. In addition it is possible to provide a function to perform the filtering. This function is expected to take the input `Chromatogram` (`object`) and to return a logical vector with the same length then there are data points in `object` with `TRUE` for data points that should be kept and `FALSE` for data points that should be removed. See examples below.

Data processing and manipulation

- `alignRt`: Aligns chromatogram `x` against chromatogram `y`. The resulting chromatogram has the same length (number of data points) than `y` and the same retention times thus allowing to perform any pair-wise comparisons between the chromatograms. If `x` is a `MChromatograms()` object, each `Chromatogram` in it is aligned against `y`. Additional parameters (`...`) are passed along to the alignment functions (e.g. `closest()`).

  Parameter `method` allows to specify which alignment method should be used. Currently there are the following options:

  - `method = "closest"` (the default): match data points in the first chromatogram (`x`) to those of the second (`y`) based on the difference between their retention times: each data point in `x` is assigned to the data point in `y` with the smallest difference in their retention times if their difference is smaller than the minimum average difference between retention times in `x` or `y` (parameter `tolerance` for the call to the `closest()` function). By setting `tolerance = 0` only exact retention times are matched against each other (i.e. only values are kept with exactly the same retention times between both chromatograms).
  - `method = "approx"`: uses the base R `approx` function to approximate intensities in `x` to the retention times in `y` (using linear interpolation). This should only be used for chromatograms that were measured in the same measurement run (e.g. MS1 and corresponding MS2 chromatograms from SWATH experiments).

- `bin`: aggregates intensity values from a chromatogram in discrete bins along the retention time axis and returns a `Chromatogram` object with the retention time representing the mid-point of the bins and the intensity the binned signal. Parameters `binSize` and `breaks` allow to define the binning, `fun` the function which should be used to aggregate the intensities within a bin.
- `compareChromatograms`: calculates a similarity score between 2 chromatograms after aligning them. Parameter `ALIGNFUN` allows to define a function that can be used to align `x` against `y` (defaults to `ALIGNFUN = alignRt`). Subsequently, the similarity is calculated on the aligned intensities with the function provided with parameter `FUN` which defaults to `cor` (hence by default the Pearson correlation is calculated between the aligned intensities of the two compared chromatograms). Additional parameters can be passed to the `ALIGNFUN` and `FUN` with the parameter `ALIGNFUNARGS` and `FUNARGS`, respectively.
- `clean`: removes 0-intensity data points (and `NA` values). See `clean()` for details.
- `normalize`, `normalise`: *normalises* the intensities of a chromatogram by dividing them either by the maximum intensity (`method = "max"`) or total intensity (`method = "sum"`) of the chromatogram.
- `transformIntensity`: allows to manipulate the intensity values of a chromatogram using a user provided function. See below for examples.

Data visualization

- `plot`: plots a `Chromatogram` object.

Author(s)

Johannes Rainer

See Also

MChromatograms for combining Chromatogram in a two-dimensional matrix (rows being mz-rt ranges, columns samples). chromatogram()] for the method to extract chromatogram data from an MSnExporOnD object.

Examples

```
## Create a simple Chromatogram object.
ints <- abs(rnorm(100, sd = 100))
rts <- seq_len(length(ints))
chr <- Chromatogram(rtime = rts, intensity = ints)
chr

## Extract intensities
intensity(chr)

## Extract retention times
rtime(chr)

## Extract the mz range - is NA for the present example
mz(chr)

## plot the Chromatogram
plot(chr)

## Create a simple Chromatogram object based on random values.
chr <- Chromatogram(intensity = abs(rnorm(1000, mean = 2000, sd = 200)),
        rtime = sort(abs(rnorm(1000, mean = 10, sd = 5))))
chr

## Get the intensities
head(intensity(chr))

## Get the retention time
head(rtime(chr))

## What is the retention time range of the object?
range(rtime(chr))

## Filter the chromatogram to keep only values between 4 and 10 seconds
chr2 <- filterRt(chr, rt = c(4, 10))

range(rtime(chr2))

## Data manipulations:

## normalize a chromatogram
par(mfrow = c(1, 2))
plot(chr)
plot(normalize(chr, method = "max"))

## Align chromatograms against each other
```

```
chr1 <- Chromatogram(rtime = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
    intensity = c(3, 5, 14, 30, 24, 6, 2, 1, 1, 0))
chr2 <- Chromatogram(rtime = c(2.5, 3.42, 4.5, 5.43, 6.5),
    intensity = c(5, 12, 15, 11, 5))

plot(chr1, col = "black")
points(rtime(chr2), intensity(chr2), col = "blue", type = "l")

## Align chr2 to chr1 without interpolation
res <- alignRt(chr2, chr1)
rtime(res)
intensity(res)
points(rtime(res), intensity(res), col = "#00ff0080", type = "l")

## Align chr2 to chr1 with interpolation
res <- alignRt(chr2, chr1, method = "approx")
points(rtime(res), intensity(res), col = "#ff000080", type = "l")
legend("topright", col = c("black", "blue", "#00ff0080","#ff000080"),lty = 1,
    legend = c("chr1", "chr2", "chr2 matchRtime", "chr2 approx"))


## Compare Chromatograms. Align chromatograms with `alignRt` and
## method `"approx"`
compareChromatograms(chr2, chr1, ALIGNFUNARGS = list(method = "approx"))

## Data filtering

chr1 <- Chromatogram(rtime = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
    intensity = c(3, 5, 14, 30, 24, 6, 2, 1, 1, 0))

## Remove data points with intensities below 10
res <- filterIntensity(chr1, 10)
intensity(res)

## Remove data points with an intensity lower than 10% of the maximum
## intensity in the Chromatogram
filt_fun <- function(x, prop = 0.1) {
    x@intensity >= max(x@intensity, na.rm = TRUE) * prop
}
res <- filterIntensity(chr1, filt_fun)
intensity(res)

## Remove data points with an intensity lower than half of the maximum
res <- filterIntensity(chr1, filt_fun, prop = 0.5)
intensity(res)

## log2 transform intensity values
res <- transformIntensity(chr1, log2)
intensity(res)
log2(intensity(chr1))
```

`chromatogram,MSnExp-method` *Extract chromatogram object(s)*

Description

The `chromatogram` method extracts chromatogram(s) from an `MSnExp` or `OnDiskMSnExp` object. Depending on the provided parameters this can be a total ion chromatogram (TIC), a base peak chromatogram (BPC) or an extracted ion chromatogram (XIC) extracted from each sample/file.

Usage

```
## S4 method for signature 'MSnExp'
chromatogram(
  object,
  rt,
  mz,
  aggregationFun = "sum",
  missing = NA_real_,
  msLevel = 1L,
  BPPARAM = bpparam()
)
```

Arguments

`object`	For `chromatogram`: a `MSnExp` or `OnDiskMSnExp` object from which the chromatogram should be extracted.
`rt`	A `numeric(2)` or two-column `matrix` defining the lower and upper boundary for the retention time range/window(s) for the chromatogram(s). If a `matrix` is provided, a chromatogram is extracted for each row. If not specified, a chromatogram representing the full retention time range is extracted. See examples below for details.
`mz`	A `numeric(2)` or two-column `matrix` defining the mass-to-charge (mz) range(s) for the chromatogram(s). For each spectrum/retention time, all intensity values within this mz range are aggregated to result in the intensity value for the spectrum/retention time. If not specified, the full mz range is considered. See examples below for details.
`aggregationFun`	`character` defining the function to be used for intensity value aggregation along the mz dimension. Allowed values are `"sum"` (TIC), `"max"` (BPC), `"min"` and `"mean"`.
`missing`	`numeric(1)` allowing to specify the intensity value for if for a given retention time (spectrum) no signal was measured within the mz range. Defaults to `NA_real_`.
`msLevel`	`integer` specifying the MS level from which the chromatogram should be extracted. Defaults to `msLevel = 1L`.

BPPARAM Parallelisation backend to be used, which will depend on the architecture. Default is BiocParallel::bpparam().

Details

Arguments rt and mz allow to specify the MS data slice from which the chromatogram should be extracted. The parameter aggregationSum allows to specify the function to be used to aggregate the intensities across the mz range for the same retention time. Setting aggregationFun = "sum" would e.g. allow to calculate the *total ion chromatogram* (TIC), aggregationFun = "max" the *base peak chromatogram* (BPC). The length of the extracted Chromatogram object, i.e. the number of available data points, corresponds to the number of scans/spectra measured in the specified retention time range. If in a specific scan (for a give retention time) no signal was measured in the specified mz range, a NA_real_ is reported as intensity for the retention time (see Notes for more information). This can be changed using the missing parameter.

By default or if \code{mz} and/or \code{rt} are numeric vectors, the function extracts one \code{\link{Chromatogram}} object for each file in the \code{\linkS4class{MSnExp}} or \code{\linkS4class{OnDiskMSnExp}} object. Providing a numeric matrix with argument \code{mz} or \code{rt} enables to extract multiple chromatograms per file, one for each row in the matrix. If the number of columns of \code{mz} or \code{rt} are not equal to 2, \code{range} is called on each row of the matrix.

Value

chromatogram returns a MChromatograms object with the number of columns corresponding to the number of files in object and number of rows the number of specified ranges (i.e. number of rows of matrices provided with arguments mz and/or rt). The featureData of the returned object contains columns "mzmin" and "mzmax" with the values from input argument mz (if used) and "rtmin" and "rtmax" if the input argument rt was used.

Author(s)

Johannes Rainer

See Also

Chromatogram and MChromatograms for the classes that represent single and multiple chromatograms.

Examples

```
## Read a test data file.
library(BiocParallel)
register(SerialParam())
library(msdata)
f <- c(system.file("microtofq/MM14.mzML", package = "msdata"),
     system.file("microtofq/MM8.mzML", package = "msdata"))

## Read the data as an MSnExp
msd <- readMSData(f, msLevel = 1)
```

```
## Extract the total ion chromatogram for each file:
tic <- chromatogram(msd)

tic

## Extract the TIC for the second file:
tic[1, 2]

## Plot the TIC for the first file
plot(rtime(tic[1, 1]), intensity(tic[1, 1]), type = "l",
    xlab = "rtime", ylab = "intensity", main = "TIC")

## Extract chromatograms for a MS data slices defined by retention time
## and mz ranges.
rtr <- rbind(c(10, 60), c(280, 300))
mzr <- rbind(c(140, 160), c(300, 320))
chrs <- chromatogram(msd, rt = rtr, mz = mzr)

## Each row of the returned MChromatograms object corresponds to one mz-rt
## range. The Chromatogram for the first range in the first file is empty,
## because the retention time range is outside of the file's rt range:
chrs[1, 1]

## The mz and/or rt ranges used are provided as featureData of the object
fData(chrs)

## The mz method can be used to extract the m/z ranges directly
mz(chrs)

## Also the Chromatogram for the second range in the second file is empty
chrs[2, 2]

## Get the extracted chromatogram for the first range in the second file
chr <- chrs[1, 2]
chr

plot(rtime(chr), intensity(chr), xlab = "rtime", ylab = "intensity")
```

---

clean-methods *Clean 'MSnExp', 'Spectrum' or 'Chromatogram' instances*

Description

This method cleans out individual spectra (`Spectrum` instances), chromatograms (`Chromatogram` instances) or whole experiments (`MSnExp` instances) of 0-intensity peaks. Unless `all` is set to `FALSE`, original 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 spectra in `MSnExp` object. Displays a control bar if verbose set to `TRUE` (default). Returns a cleaned `MSnExp` instance.

`signature(object = "Spectrum", all = "logical", msLevel. = "numeric")` Cleans the `Spectrum` object. Returns a cleaned `Spectrum` instance. If `all = TRUE`, then all zeros are removed. `msLevel.` defines the level of the spectrum, and if `msLevel(object) != msLevel.`, cleaning is ignored. Only relevant when called from `OnDiskMSnExp` and is only relevant for developers.

`signature(object = "Chromatogram", all = "logical", na.rm = "logical")` Cleans the `Chromatogram` instance and returns a cleaned `Chromatogram` object. If `na.rm` is `TRUE` (default is `FALSE`) all `NA` intensities are removed before cleaning the chromatogram.

Author(s)

Laurent Gatto

See Also

`removePeaks` and `trimMz` for other spectra processing methods.

Examples

```
int <- c(1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0)
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)

## Create a simple Chromatogram object
chr <- Chromatogram(rtime = 1:12,
                    intensity = c(0, 0, 20, 0, 0, 0, 123, 124343, 3432, 0, 0, 0))

## Remove 0-intensity values keeping those adjacent to peaks
chr <- clean(chr)
intensity(chr)

## Remove all 0-intensity values
```

```
chr <- clean(chr, all = TRUE)
intensity(chr)

## Clean a Chromatogram with NAs.
chr <- Chromatogram(rtime = 1:12,
                    intensity = c(0, 0, 20, NA, NA, 0, 123, 124343, 3432, 0, 0, 0))
chr <- clean(chr, all = FALSE, na.rm = TRUE)
intensity(chr)
```

`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 `fcol` 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.

If NA values are present, a message will be shown. Details on how missing value impact on the data aggregation are provided below.

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 summerise 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`	Deprecated; use `method` instead.
`method`	The summerising function. Currently, mean, median, weighted mean, sum, median polish, robust summarisation (using `MASS::rlm`, implemented in `MsCoreUtils::robustSummary()`) iPQF (see `iPQF` for details) and NTR (see `NTR` for details) are implemented, but user-defined functions can also be supplied. Note that the robust menthods assumes that the data are already log-transformed.
`fcol`	Feature meta-data label (fData column name) defining how to summerise the features. It must be present in `fvarLabels(object)` and, if present, will be used to defined `groupBy` as `fData(object)[, fcol]`. Note that `fcol` is ignored if `groupBy` is present.
`redundancy.handler`	If `groupBy` is a `list`, one of "unique" (default) or "multiple" (ignored otherwise) 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 discarded). "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 calculation. 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.

Details

Missing values have different effect based on the aggregation method employed, as detailed below. See also examples below.

1. When using either "sum", "mean", "weighted.mean" or "median", any missing value will be propagated at the higher level. If na.rm = TRUE is used, then the missing value will be ignored.
2. Missing values will result in an error when using "medpolish", unless na.rm = TRUE is used.
3. When using robust summarisation ("robust"), individual missing values are excluded prior to fitting the linear model by robust regression. To remove all values in the feature containing the missing values, use filterNA.
4. The "iPQF" method will fail with an error if missing value are present, which will have to be handled explicitly. See below.

More generally, missing values often need dedicated handling such as filtering (see filterNA) or imputation (see impute).

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 with contributions from Martina Fischer for iPQF and Ludger Goeminne, Adriaan Sticker and Lieven Clement for robust.

References

iPQF: a new peptide-to-protein summarization method using peptide spectra characteristics to improve protein quantification. Fischer M, Renard BY. Bioinformatics. 2016 Apr 1;32(7):1040-7. doi:10.1093/bioinformatics/btv675. Epub 2015 Nov 20. PubMed PMID:26589272.

See Also

featureCV to calculate coefficient of variation, nFeatures to document the number of features per group in the feature data, and the aggvar to explore variability within protein groups.

iPQF for iPQF summarisation.

NTR for normalisation to reference summarisation.

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, groupBy = grp, method = "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, groupBy = grpl, method = "sum", redundancy.handler = "unique"))
exprs(combineFeatures(msnset, groupBy = grpl, method = "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)
anyNA(msnset2)
res <- combineFeatures(msnset2,
                       groupBy = fData(msnset2)$accession,
                       redundancy.handler = "unique",
                       method = "iPQF",
                       low.support.filter = FALSE,
                       ratio.calc = "sum",
                       method.combine = FALSE)
head(exprs(res))

## using robust summarisation
data(msnset) ## reset data
msnset <- log(msnset, 2) ## log2 transform
```

```
## Feature X46, in the ENO protein has one missig value
which(is.na(msnset), arr.ind = TRUE)
exprs(msnset["X46", ])
## Only the missing value in X46 and iTRAQ4.116 will be ignored
res <- combineFeatures(msnset,
                       fcol = "ProteinAccession",
                       method = "robust")
tail(exprs(res))

msnset2 <- filterNA(msnset) ## remove features with missing value(s)
res2 <- combineFeatures(msnset2,
                        fcol = "ProteinAccession",
                        method = "robust")
## Here, the values for ENO are different because the whole feature
## X46 that contained the missing value was removed prior to fitting.
tail(exprs(res2))
```

`combineSpectra,MSnExp-method`
Combine Spectra

Description

`combineSpectra` combines spectra in a MSnExp, OnDiskMSnExp or MSpectra object applying the summarization function `fun` to sets of spectra defined by a factor (`fcol` parameter). The resulting combined spectrum for each set contains metadata information (present in `mcols` and all spectrum information other than `mz` and `intensity`) from the **first** spectrum in each set.

Combining of spectra for MSnExp or OnDiskMSnExp objects is performed by default for each file **separately**, combining of spectra across files is thus not possible. See examples for details.

Usage

```
## S4 method for signature 'MSnExp'
combineSpectra(
  object,
  fcol = "fileIdx",
  method = meanMzInts,
  ...,
  BPPARAM = bpparam()
)

## S4 method for signature 'MSpectra'
combineSpectra(object, fcol, method = meanMzInts, ...)
```

Arguments

`object`	A MSnExp or MSpectra

fcol	For MSpectra objects: mcols column name to be used to define the sets of spectra to be combined. If missing, all spectra are considered to be one set. For MSnExp/OnDiskMSnExp objects: column in fData(object) defining which spectra to combine. See examples below for more details.
method	function to be used to combine the spectra by fcol. Has to be a function that takes a list of spectra as input and returns a single Spectrum. See meanMzInts() for details.
...	additional arguments for fun.
BPPARAM	For MSnExp/OnDiskMSnExp objects: parallel processing setup to perform per-file parallel spectra combining. See bpparam() for more details.

Value

A MSpectra or MSnExp object with combined spectra. Metadata (mcols) and all spectrum attributes other than mz and intensity are taken from the first Spectrum in each set.

Author(s)

Johannes Rainer, Laurent Gatto

See Also

meanMzInts() for a function to combine spectra.

Examples

```
set.seed(123)
mzs <- seq(1, 20, 0.1)
ints1 <- abs(rnorm(length(mzs), 10))
ints1[11:20] <- c(15, 30, 90, 200, 500, 300, 100, 70, 40, 20) # add peak
ints2 <- abs(rnorm(length(mzs), 10))
ints2[11:20] <- c(15, 30, 60, 120, 300, 200, 90, 60, 30, 23)
ints3 <- abs(rnorm(length(mzs), 10))
ints3[11:20] <- c(13, 20, 50, 100, 200, 100, 80, 40, 30, 20)

## Create the spectra.
sp1 <- new("Spectrum1", mz = mzs + rnorm(length(mzs), sd = 0.01),
    intensity = ints1, rt = 1)
sp2 <- new("Spectrum1", mz = mzs + rnorm(length(mzs), sd = 0.01),
    intensity = ints2, rt = 2)
sp3 <- new("Spectrum1", mz = mzs + rnorm(length(mzs), sd = 0.009),
    intensity = ints3, rt = 3)

spctra <- MSpectra(sp1, sp2, sp3,
    elementMetadata = DataFrame(idx = 1:3, group = c("b", "a", "a")))

## Combine the spectra reporting the maximym signal
res <- combineSpectra(spctra, mzd = 0.05, intensityFun = max)
res

## All values other than m/z and intensity are kept from the first spectrum
```

```
rtime(res)

## Plot the individual and the merged spectrum
par(mfrow = c(2, 1), mar = c(4.3, 4, 1, 1))
plot(mz(sp1), intensity(sp1), xlim = range(mzs[5:25]), type = "h", col = "red")
points(mz(sp2), intensity(sp2), type = "h", col = "green")
points(mz(sp3), intensity(sp3), type = "h", col = "blue")
plot(mz(res[[1]]), intensity(res[[1]]), type = "h",
    col = "black", xlim = range(mzs[5:25]))

## Combine spectra in two sets.
res <- combineSpectra(spctra, fcol = "group", mzd = 0.05)
res

rtime(res)

## Plot the individual and the merged spectra
par(mfrow = c(3, 1), mar = c(4.3, 4, 1, 1))
plot(mz(sp1), intensity(sp1), xlim = range(mzs[5:25]), type = "h", col = "red")
points(mz(sp2), intensity(sp2), type = "h", col = "green")
points(mz(sp3), intensity(sp3), type = "h", col = "blue")
plot(mz(res[[1]]), intensity(res[[1]]), xlim = range(mzs[5:25]), type = "h",
    col = "black")
plot(mz(res[[2]]), intensity(res[[2]]), xlim = range(mzs[5:25]), type = "h",
    col = "black")

## Combining spectra of an MSnExp/OnDiskMSnExp objects
## Reading data from 2 mzML files
sciex <- readMSData(dir(system.file("sciex", package = "msdata"),
    full.names = TRUE), mode = "onDisk")

## Filter the file to a retention time range from 2 to 20 seconds (to reduce
## execution time of the example)
sciex <- filterRt(sciex, rt = c(2, 20))
table(fromFile(sciex))

## We have thus 64 spectra per file.

## In the example below we combine spectra measured in one second to a
## single spectrum. We thus first define the grouping variable and add that
## to the `fData` of the object. For combining, we use the
## `consensusSpectrum` function that combines the spectra keeping only peaks
## that were found in 50% of the spectra; by defining `mzd = 0.01` all peaks
## within an m/z of 0.01 are evaluated for combining.
seconds <- round(rtime(sciex))
head(seconds)
fData(sciex)$second <- seconds

res <- combineSpectra(sciex, fcol = "second", mzd = 0.01, minProp = 0.1,
    method = consensusSpectrum)
table(fromFile(res))

## The data was reduced to 19 spectra for each file.
```

combineSpectraMovingWindow *Combine signal from consecutive spectra of LCMS experiments*

Description

combineSpectraMovingWindow combines signal from consecutive spectra within a file. The resulting MSnExp has the same total number of spectra than the original object, but with each individual's spectrum information representing aggregated data from the original spectrum and its neighboring spectra. This is thus equivalent with a smoothing of the data in retention time dimension.

Note that the function returns always a MSnExp object, even if x was an OnDiskMSnExp object.

Usage

```
combineSpectraMovingWindow(
  x,
  halfWindowSize = 1L,
  intensityFun = base::mean,
  mzd = NULL,
  timeDomain = FALSE,
  weighted = FALSE,
  ppm = 0,
  BPPARAM = bpparam()
)
```

Arguments

x	MSnExp or OnDiskMSnExp object.
halfWindowSize	integer(1) with the half window size for the moving window.
intensityFun	function to aggregate the intensity values per m/z group. Should be a function or the name of a function. The function is expected to return a numeric(1).
mzd	numeric(1) defining the maximal m/z difference below which mass peaks are considered to represent the same ion/mass peak. Intensity values for such grouped mass peaks are aggregated. If not specified this value is estimated from the distribution of differences of m/z values from the provided spectra (see details).
timeDomain	logical(1) whether definition of the m/z values to be combined into one m/z is performed on m/z values (timeDomain = FALSE) or on sqrt(mz) (timeDomain = TRUE). Profile data from TOF MS instruments should be aggregated based on the time domain (see details). Note that a pre-defined mzd should also be estimated on the square root of m/z values if timeDomain = TRUE.
weighted	logical(1) whether m/z values per m/z group should be aggregated with an intensity-weighted mean. The default is to report the mean m/z.
ppm	numeric(1) to define an m/z relative deviation. Note that if only ppm should be considered but not mzd, mzd should be set to 0 (i.e. mzd = 0). This parameter is directly passed to meanMzInts().
BPPARAM	parallel processing settings.

Details

The method assumes same ions being measured in consecutive scans (i.e. LCMS data) and thus combines their signal which can increase the increase the signal to noise ratio.

Intensities (and m/z values) for signals with the same m/z value in consecutive scans are aggregated using the `intensityFun`. m/z values of intensities from consecutive scans will never be exactly identical, even if they represent signal from the same ion. The function determines thus internally a similarity threshold based on differences between m/z values within and between spectra below which m/z values are considered to derive from the same ion. For robustness reasons, this threshold is estimated on the 100 spectra with the largest number of m/z - intensity pairs (i.e. mass peaks).

See `meanMzInts()` for details.

Parameter `timeDomain`: by default, m/z-intensity pairs from consecutive scans to be aggregated are defined based on the square root of the m/z values. This is because it is highly likely that in all QTOF MS instruments data is collected based on a timing circuit (with a certain variance) and m/z values are later derived based on the relationship `t = k * sqrt(m/z)`. Differences between individual m/z values will thus be dependent on the actual m/z value causing both the difference between m/z values and their scattering being different in the lower and upper m/z range. Determining m/z values to be combined on the `sqrt(mz)` reduces this dependency. For non-QTOF MS data `timeDomain = FALSE` might be used instead.

Value

`MSnExp` with the same number of spectra than `x`.

Note

The function has to read all data into memory for the spectra combining and thus the memory requirements of this function are high, possibly preventing its usage on large experimental data. In these cases it is suggested to perform the combination on a per-file basis and save the results using the `writeMSData()` function afterwards.

Author(s)

Johannes Rainer, Sigurdur Smarason

See Also

`meanMzInts()` for the function combining spectra provided in a `list`.

`estimateMzScattering()` for a function to estimate m/z value scattering in consecutive spectra.

Examples

```
library(MSnbase)
library(msdata)

## Read a profile-mode LC-MS data file.
fl <- dir(system.file("sciex", package = "msdata"), full.names = TRUE)[1]
od <- readMSData(fl, mode = "onDisk")
```

```
## Subset the object to the retention time range that includes the signal
## for proline. This is done for performance reasons.
rtr <- c(165, 175)
od <- filterRt(od, rtr)

## Combine signal from neighboring spectra.
od_comb <- combineSpectraMovingWindow(od)

## The combined spectra have the same number of spectra, same number of
## mass peaks per spectra, but the signal is larger in the combined object.
length(od)
length(od_comb)

peaksCount(od)
peaksCount(od_comb)

## Comparing the chromatographic signal for proline (m/z ~ 116.0706)
## before and after spectra data combination.
mzr <- c(116.065, 116.075)
chr <- chromatogram(od, rt = rtr, mz = mzr)
chr_comb <- chromatogram(od_comb, rt = rtr, mz = mzr)

par(mfrow = c(1, 2))
plot(chr)
plot(chr_comb)
## Chromatographic data is "smoother" after combining.
```

`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` or `MSnSetList` with at least 2 `MSnSet` objects.
`y`	An instance of class `MSnSet`. Ignored if `x` is a `list`/`MSnSetList`.

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(x = "MSnExp", y = "missing", fun = "character", ...)` Compares all spectra in an `MSnExp` object. The `...` arguments are passed to the internal functions. Returns a `matrix` of dimension `length(x)` by `length(x)`.

`signature(x = "Spectrum", y = "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 algorithms for compound identification. Journal of the American Society for Mass Spectrometry, 5(9), 859-866. doi: https://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: https://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")
```

---

consensusSpectrum *Combine spectra to a consensus spectrum*

Description

consensusSpectrum takes a list of spectra and combines them to a consensus spectrum containing mass peaks that are present in a user definable proportion of spectra.

Usage

```
consensusSpectrum(
  x,
  mzd = 0,
  minProp = 0.5,
  intensityFun = stats::median,
  mzFun = stats::median,
  ppm = 0,
  weighted = FALSE,
  ...
)
```

Arguments

x	list of Spectrum objects (either Spectrum1 or Spectrum2).
mzd	numeric(1) defining the maximal m/z difference below which mass peaks are grouped in to the same final mass peak (see details for more information). Defaults to 0; see meanMzInts() for estimating this value from the distribution of differences of m/z values from the spectra. See also parameter ppm below for the definition of an m/z dependent peak grouping.

minProp	numeric(1) defining the minimal proportion of spectra in which a mass peak has to be present in order to include it in the final consensus spectrum. Should be a number between 0 and 1 (present in all spectra).
intensityFun	function (or name of a function) to be used to define the intensity of the aggregated peak. By default the median signal for a mass peak is reported.
mzFun	function (or name of a function) to be used to define the intensity of the aggregated peak. By default the median m/z is reported. Note that setting weighted = TRUE overrides this parameter.
ppm	numeric(1) allowing to perform a m/z dependent grouping of mass peaks. See details for more information.
weighted	logical(1) whether the m/z of the aggregated peak represents the intensity-weighted average of the m/z values of all peaks of the peak group. If FALSE (the default), the m/z of the peak is calculated with mzFun.
...	additional arguments to be passed to intensityFun.

Details

Peaks from spectra with a difference of their m/z being smaller than mzd are grouped into the same final mass peak with their intensities being aggregated with intensityFun. Alternatively (or in addition) it is possible to perform an m/z dependent grouping of mass peaks with parameter ppm: mass peaks from different spectra with a difference in their m/z smaller than ppm of their m/z are grouped into the same final peak.

The m/z of the final mass peaks is calculated with mzFun. By setting weighted = TRUE the parameter mzFun is ignored and an intensity-weighted mean of the m/z values from the individual mass peaks is returned as the peak's m/z.

Author(s)

Johannes Rainer

See Also

Other spectra combination functions: meanMzInts()

Examples

```
library(MSnbase)
## Create 3 example spectra.
sp1 <- new("Spectrum2", rt = 1, precursorMz = 1.41,
    mz = c(1.2, 1.5, 1.8, 3.6, 4.9, 5.0, 7.8, 8.4),
    intensity = c(10, 3, 140, 14, 299, 12, 49, 20))
sp2 <- new("Spectrum2", rt = 1.1, precursorMz = 1.4102,
    mz = c(1.4, 1.81, 2.4, 4.91, 6.0, 7.2, 9),
    intensity = c(3, 184, 8, 156, 12, 23, 10))
sp3 <- new("Spectrum2", rt = 1.2, precursorMz = 1.409,
    mz = c(1, 1.82, 2.2, 3, 7.0, 8),
    intensity = c(8, 210, 7, 101, 17, 8))
spl <- MSpectra(sp1, sp2, sp3)
```

```
## Plot the spectra, each in a different color
par(mfrow = c(2, 1), mar = c(4.3, 4, 1, 1))
plot(mz(sp1), intensity(sp1), type = "h", col = "#ff000080", lwd = 2,
    xlab = "m/z", ylab = "intensity", xlim = range(mz(spl)),
    ylim = range(intensity(spl)))
points(mz(sp2), intensity(sp2), type = "h", col = "#00ff0080", lwd = 2)
points(mz(sp3), intensity(sp3), type = "h", col = "#0000ff80", lwd = 2)

cons <- consensusSpectrum(spl, mzd = 0.02, minProp = 2/3)

## Peaks of the consensus spectrum
mz(cons)
intensity(cons)

## Other Spectrum data is taken from the first Spectrum in the list
rtime(cons)
precursorMz(cons)

plot(mz(cons), intensity(cons), type = "h", xlab = "m/z", ylab = "intensity",
    xlim = range(mz(spl)), ylim = range(intensity(spl)), lwd = 2)
```

`Deprecated` *MSnbase Deprecated and Defunct*

Description

The function, class, or data object you have asked for has been deprecated or made defunct.

Deprecated:

Defunct: `readMzXMLData`, `extractSpectra`, `writeMzTabData`, `makeMTD`, `makePEP`, `makePRT`, `NAnnotatedDataFrame` class.

`estimateMzResolution,MSnExp-method`
Estimate the m/z resolution of a spectrum

Description

`estimateMzResolution` estimates the m/z resolution of a profile-mode `Spectrum` (or of all spectra in an MSnExp or OnDiskMSnExp object. The m/z resolution is defined as the most frequent difference between a spectrum's m/z values.

Usage

```
## S4 method for signature 'MSnExp'
estimateMzResolution(object, ...)

## S4 method for signature 'Spectrum'
estimateMzResolution(object, ...)
```

Arguments

object	either a Spectrum, MSnExp or OnDiskMSnExp object.
...	currently not used.

Value

numeric(1) with the m/z resolution. If called on a MSnExp or OnDiskMSnExp a list of m/z resolutions are returned (one for each spectrum).

Note

This assumes the data to be in profile mode and does not return meaningful results for centroided data.

The estimated m/z resolution depends on the number of ions detected in a spectrum, as some instrument don't measure (or report) signal if below a certain threshold.

Author(s)

Johannes Rainer

Examples

```
## Not run:
## Load a profile mode example file
library(BiocParallel)
register(SerialParam())

f <- MsDataHub::TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.20141210.mzML.gz()
od <- readMSData(f, mode = "onDisk")

## Estimate the m/z resolution on the 3rd spectrum.
estimateMzResolution(od[[3]])

## Estimate the m/z resolution for each spectrum
mzr <- estimateMzResolution(od)

## plot the distribution of estimated m/z resolutions. The bimodal
## distribution represents the m/z resolution of the MS1 (first peak) and
## MS2 spectra (second peak).
plot(density(unlist(mzr)))

## End(Not run)
```

`estimateMzScattering` *Estimate m/z scattering in consecutive scans*

Description

Estimate scattering of m/z values (due to technical, instrument specific noise) for the same ion in consecutive scans of a LCMS experiment.

Usage

```
estimateMzScattering(x, halfWindowSize = 1L, timeDomain = FALSE)
```

Arguments

`x`	`MSnExp` or `OnDiskMSnExp` object.
`halfWindowSize`	`integer(1)` defining the half window size for the moving window to combine consecutive spectra.
`timeDomain`	`logical(1)` whether m/z scattering should be estimated on `mz` (`timeDomain = FALSE`) or `sqrt(mz)` (`timeDomain = TRUE`) values. See `combineSpectraMovingWindow()` for details on this parameter.

Details

The m/z values of the same ions in consecutive scans (spectra) of a LCMS run will not be identical. This random noise is expected to be smaller than the resolution of the MS instrument. The distribution of differences of m/z values from neighboring spectra is thus expected to be (at least) bi-modal with the first peak representing the above described random variation and the second (or largest) peak the m/z resolution. The m/z value of the first local minimum between these first two peaks in the distribution is returned as the *m/z scattering*.

Note

For `timeDomain = TRUE` the function does **not** return the estimated scattering of m/z values, but the scattering of `sqrt(mz)` values.

Author(s)

Johannes Rainer

See Also

`estimateMzResolution()` for the function to estimate a profile-mode spectrum's m/z resolution from it's data.

Examples

```
library(MSnbase)
library(msdata)
## Load a profile-mode LC-MS data file
f <- dir(system.file("sciex", package = "msdata"), full.names = TRUE)[1]
od <- readMSData(f, mode = "onDisk")
im <- as(filterRt(od, c(10, 20)), "MSnExp")

res <- estimateMzScattering(im)

## Plot the distribution of estimated m/z scattering
plot(density(unlist(res)))

## Compare the m/z resolution and m/z scattering of the spectrum with the
## most peaks
idx <- which.max(unlist(spectrapply(im, peaksCount)))

res[[idx]]
abline(v = res[[idx]], lty = 2)
estimateMzResolution(im[[idx]])
## As expected, the m/z scattering is much lower than the m/z resolution.
```

`estimateNoise-methods` *Noise Estimation for 'Spectrum' instances*

Description

This method performs a noise estimation on individual spectra (`Spectrum` instances). 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(object = "Spectrum", method = "character", ...)` Estiamtes the noise in a non-centroided spectrum (`Spectrum` instance). `method` could be `"MAD"` or `"SuperSmoother"`. The arguments `...` are passed to the noise estimator functions implemented in `MALDIquant::estimateNoise`. Currenlty only the `method = "SuperSmoother"` accepts additional arguments, e.g. `span`. Please see `supsmu` for details. This method returns a two-column matrix with the m/z and intensity values in the first and the second column.

`signature(object = "MSnExp", method = "character", ...)` Estimates noise for all spectra in `object`.

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

pickPeaks, and the underlying method in MALDIquant: estimateNoise.

Examples

```
sp1 <- new("Spectrum1",
           intensity = c(1:6, 5:1),
           mz = 1:11,
           centroided = FALSE)
estimateNoise(sp1, method = "SuperSmoother")
```

---

expandFeatureVars *Expand or merge feature variables*

Description

The expandFeatureVars and mergeFeatureVars respectively expand and merge groups of feature variables. Using these functions, a set of columns in a feature data can be merged into a single new data.frame-column variables and a data.frame-column can be expanded into single feature columns. The original feature variables are removed.

Usage

```
expandFeatureVars(x, fcol, prefix)

mergeFeatureVars(x, fcol, fcol2)
```

Arguments

x	An object of class MSnSet.
fcol	A character() of feature variables to expand (for expandFeatureVars) or merge (for mergeFeatureVars).
prefix	A character(1) to use as prefix to the new feature variables. If missing (default), then fcol is used instead. If NULL, then no prefix is used.
fcol2	A character(1) defining the name of the new feature variable.

Value

An MSnSet for expanded (merged) feature variables.

Author(s)

Laurent Gatto

Examples

```
library("pRolocdata")
data(hyperLOPIT2015)
fvarLabels(hyperLOPIT2015)
## Let's merge all svm prediction feature variables
(k <- grep("^svm", fvarLabels(hyperLOPIT2015), value = TRUE))
hl <- mergeFeatureVars(hyperLOPIT2015, fcol = k, fcol2 = "SVM")
fvarLabels(hl)
head(fData(hl)$SVM)

## Let's expand the new SVM into individual columns
hl2 <- expandFeatureVars(hl, "SVM")
fvarLabels(hl2)
## We can set the prefix manually
hl2 <- expandFeatureVars(hl, "SVM", prefix = "Expanded")
fvarLabels(hl2)
## If we don't want any prefix
hl2 <- expandFeatureVars(hl, "SVM", prefix = NULL)
fvarLabels(hl2)
```

---

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 the`prec` 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

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)
```

`extractSpectraData` *Conversion between objects from the Spectra and MSnbase packages*

Description

The Spectra package provides a more robust and efficient infrastructure for mass spectrometry data handling and analysis. So, wherever possible, the newer *Spectra* package should be used instead of the *MSnbase*. The functions listed here allow to convert between objects from the *MSnbase* and *Spectra* packages.

`extractSpectraData` extracts the spectra data (m/z and intensity values including metadata) from MSnExp, OnDiskMSnExp, Spectrum1, Spectrum2 objects (or `list` of such objects) and returns these as a `DataFrame` that can be used to create a Spectra::Spectra object.This function enables thus to convert data from the *old* `MSnbase` package to the newer `Spectra` package.

To convert a `Spectra` object to a `MSpectra` object use `as(sps, "MSpectra")` where `sps` is a `Spectra` object.

Usage

```
extractSpectraData(x)
```

Arguments

`x`	a `list` of Spectrum objects or an object extending MSnExp or a MSpectra object.

Value

- `extracSpectraData()` returns a `DataFrame()` with the full spectrum data that can be passed to the `Spectra::Spectra()` function to create a `Spectra` object.
- `as(x, "MSpectra")` returns a `MSpectra` object with the content of the `Spectra` object `x`.

Note

Coercion from `Spectra` to a `MSpectra` will only assign values to the contained `Spectrum1` and `Spectrum2` objects, but will not add all eventually spectra variables present in `Spectra`.

Author(s)

Johannes Rainer

Examples

```
## Read an mzML file with MSnbase
fl <- MsDataHub::PestMix1_SWATH.mzML()
data <- filterRt(readMSData(fl, mode = "onDisk"), rt = c(1, 6))

## Extract the data as a DataFrame
```

```
res <- extractSpectraData(data)
res

library(Spectra)
## This can be used as an input for the Spectra constructor of the
## Spectra package:
sps <- Spectra::Spectra(res)
sps

## A Spectra object can be coerced to a MSnbase MSpectra object using
msps <- as(sps, "MSpectra")
```

`factorsAsStrings` *Converts factors to strings*

Description

This function produces the opposite as the `stringsAsFactors` argument in the `data.frame` or `read.table` functions; it converts `factors` columns to `characters`.

Usage

```
factorsAsStrings(x)
```

Arguments

`x`	A `data.frame`

Value

A `data.frame` where `factors` are converted to `characters`.

Author(s)

Laurent Gatto

Examples

```
data(iris)
str(iris)
str(factorsAsStrings(iris))
```

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 = "logical", verbose = "logical"): 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 and Thomas Naake

See Also

averageMSnSet to compuate 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 = "none", suffix = NULL)
```

Arguments

x	An instance of class MSnSet.
groupBy	An object of class factor defining how to summarise the features.
na.rm	A logical(1) defining whether missing values should be removed.
norm	One of normalisation methods applied prior to CV calculation. See normalise() for more details. Here, the default is 'none', i.e. no normalisation.

suffix — A character(1) to be used to name the new CV columns. Default is NULL to ignore this. This argument should be set when CV values are already present in the MSnSet feature variables.

Value

A matrix of dimensions length(levels(groupBy)) by ncol(x) with the respecive CVs. The column names are formed by pasting CV. and the sample names of object x, possibly suffixed by .suffix.

Author(s)

Laurent Gatto and Sebastian Gibb

See Also

combineFeatures()

Examples

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

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.

[Subsetting works like lists. Returns a new `FoICollection`.

[[Subsetting works like lists. Returns a new `FeatureOfInterest`.

`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 `i`th `FeatureOfInterest` in the collection `object`.

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

Author(s)

Laurent Gatto

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)
names(xx) <- LETTERS[1:2]
```

```
xx

## Sub-setting
xx[1]
xx[[1]]
xx[["A"]]

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 `NA`s 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

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])
```

`filterIdentificationDataFrame` *Filter out unreliable PSMs.*

Description

A function to filter out PSMs matching to the decoy database, of rank greater than one and matching non-proteotypic peptides.

Usage

```
filterIdentificationDataFrame(
  x,
  decoy = "isDecoy",
  rank = "rank",
  accession = "DatabaseAccess",
  spectrumID = "spectrumID",
  verbose = isMSnbaseVerbose()
)
```

Arguments

`x`	A `data.frame` containing PSMs.
`decoy`	The column name defining whether entries match the decoy database. Default is `"isDecoy"`. The column should be a `logical` and only PSMs holding a `FALSE` are retained. Ignored is set to `NULL`.
`rank`	The column name holding the rank of the PSM. Default is `"rank"`. This column should be a `numeric` and only PSMs having rank equal to 1 are retained. Ignored is set to `NULL`.
`accession`	The column name holding the protein (groups) accession. Default is `"DatabaseAccess"`. Ignored is set to `NULL`.
`spectrumID`	The name of the spectrum identifier column. Default is `spectrumID`.
`verbose`	A `logical` verbosity flag. Default is to take `isMSnbaseVerbose()`.

Details

The PSMs should be stored in a `data.frame` such as those produced by `readMzIdData()`. Note that this function should be called before calling the reduce method on a PSM `data.frame`.

Value

A new `data.frame` with filtered out peptides and with the same columns as the input `x`.

Author(s)

Laurent Gatto

`formatRt` *Format Retention Time*

Description

This function is used to convert retention times. Conversion is seconds to/from the more human friendly format "mm:sec". The implementation is from `MsCoreUtils::formatRt()`.

Usage

```
formatRt(rt)
```

Arguments

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

Value

A vector of same length as `rt`.

Author(s)

Laurent Gatto and Sebastian Gibb

Examples

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

`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.

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.

Usage

```
grepEcols(f, pattern, ..., n = 1)

getEcols(f, ..., n = 1)
```

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.
n	An integer specifying which line in file f to grep (get). Default is 1. Note that this argument must be named.

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`

`hasSpectra` *Checks if raw data files have any spectra or chromatograms*

Description

Helper functions to check whether raw files contain spectra or chromatograms.

Usage

```
hasSpectra(files)

hasChromatograms(files)
```

Arguments

`files`	A `character()` with raw data filenames.

Value

A `logical(n)` where `n == length(x)` with `TRUE` if that files contains at least one spectrum, `FALSE` otherwise.

Author(s)

Laurent Gatto

Examples

```
f <- msdata::proteomics(full.names = TRUE)[1:2]
hasSpectra(f)
hasChromatograms(f)
```

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,MSnSet-method *Quantitative proteomics data imputation*

Description

The `impute` method performs data imputation on `MSnSet` instances using a variety of methods.

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.

See `MsCoreUtils::impute_matrix()` for details on the different imputation methods available and strategies.

Usage

```
## S4 method for signature 'MSnSet'
impute(object, method, ...)
```

Arguments

`object`	An `MSnSet` object with missing values to be imputed.
`method`	`character(1)` defining the imputation method. See `MsCoreUtils::imputeMethods()` for available ones. See `MsCoreUtils::impute_matrix()` for details.
`...`	Additional parameters passed to the inner imputation function. See `MsCoreUtils::impute_matrix()` for details.

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 spectra characteristics to improve protein quantification. Fischer M, Renard BY. Bioinformatics. 2016 Apr 1;32(7):1040-7. doi:10.1093/bioinformatics/btv675. Epub 2015 Nov 20. PubMed PMID:26589272.

Examples

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

`isCentroidedFromFile` *Get mode from mzML data file*

Description

The function extracts the mode (profile or centroided) from the raw mass spectrometry file by parsing the mzML file directly. If the object `x` stems from any other type of file, `NA`s are returned.

Usage

```
isCentroidedFromFile(x)
```

Arguments

`x`	An object of class OnDiskMSnExp.

Details

This function is much faster than `isCentroided()`, which estimates mode from the data, but is limited to data stemming from mzML files which are still available in their original location (and accessed with `fileNames(x)`).

Value

A named `logical` vector of the same length as `x`.

Author(s)

Laurent Gatto

Examples

```
## Not run:
f <- MsDataHub::TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.20141210.mzML.gz()
x <- readMSData(f, mode = "onDisk")
table(isCentroidedFromFile(x), msLevel(x))

## End(Not run)
```

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 *and* MSnSet *data sets*

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 list.
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")
```

makeCamelCase *Convert to camel case by replacing dots by captial letters*

Description

Convert a vector of characters to camel case by replacing dots by captial letters.

Usage

```
makeCamelCase(x, prefix)
```

Arguments

x	A vector to be transformed to camel case.
prefix	An optional character of length one. Any additional elements are ignores.

Value

A `character` of same length as `x`.

Author(s)

Laurent Gatto

Examples

```
nms <- c("aa.foo", "ab.bar")
makeCamelCase(nms)
makeCamelCase(nms, prefix = "x")
```

`makeNaData` *Create a data with missing values*

Description

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

Usage

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

makeNaData2(object, nRows, nNAs, exclude)

whichNA(x)
```

Arguments

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

Details

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

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

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

Value

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

Author(s)

Laurent Gatto

Examples

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

```
(res3 <- table(fData(x3)$nNA))
```

MChromatograms *Container for multiple Chromatogram objects*

Description

The MChromatograms class allows to store Chromatogram() objects in a matrix-like two-dimensional structure.

Usage

```
MChromatograms(data, phenoData, featureData, ...)

## S4 method for signature 'MChromatograms'
show(object)

## S4 method for signature 'MChromatograms,ANY,ANY,ANY'
x[i, j, drop = FALSE]

## S4 replacement method for signature 'MChromatograms,ANY,ANY,ANY'
x[i, j] <- value

## S4 method for signature 'MChromatograms,ANY'
plot(
  x,
  col = "#00000060",
  lty = 1,
  type = "l",
  xlab = "retention time",
  ylab = "intensity",
  main = NULL,
  ...
)

## S4 method for signature 'MChromatograms'
phenoData(object)

## S4 method for signature 'MChromatograms'
pData(object)

## S4 replacement method for signature 'MChromatograms,data.frame'
pData(object) <- value

## S4 method for signature 'MChromatograms'
x$name
```

```
## S4 replacement method for signature 'MChromatograms'
x$name <- value

## S4 replacement method for signature 'MChromatograms,ANY'
colnames(x) <- value

## S4 method for signature 'MChromatograms'
sampleNames(object)

## S4 replacement method for signature 'MChromatograms,ANY'
sampleNames(object) <- value

## S4 method for signature 'MChromatograms'
isEmpty(x)

## S4 method for signature 'MChromatograms'
featureNames(object)

## S4 replacement method for signature 'MChromatograms'
featureNames(object) <- value

## S4 method for signature 'MChromatograms'
featureData(object)

## S4 replacement method for signature 'MChromatograms,ANY'
featureData(object) <- value

## S4 method for signature 'MChromatograms'
fData(object)

## S4 replacement method for signature 'MChromatograms,ANY'
fData(object) <- value

## S4 method for signature 'MChromatograms'
fvarLabels(object)

## S4 replacement method for signature 'MChromatograms'
rownames(x) <- value

## S4 method for signature 'MChromatograms'
precursorMz(object)

## S4 method for signature 'MChromatograms'
productMz(object)

## S4 method for signature 'MChromatograms'
mz(object)
```

```
## S4 method for signature 'MChromatograms'
polarity(object)

## S4 method for signature 'MChromatograms'
bin(x, binSize = 0.5, breaks = numeric(), fun = max)

## S4 method for signature 'MChromatograms'
clean(object, all = FALSE, na.rm = FALSE)

## S4 method for signature 'MChromatograms'
normalize(object, method = c("max", "sum"))

## S4 method for signature 'MChromatograms'
filterIntensity(object, intensity = 0, ...)

## S4 method for signature 'MChromatograms,Chromatogram'
alignRt(x, y, method = c("closest", "approx"), ...)

## S4 method for signature 'MChromatograms'
c(x, ...)

## S4 method for signature 'MChromatograms,missing'
compareChromatograms(
  x,
  y,
  ALIGNFUN = alignRt,
  ALIGNFUNARGS = list(),
  FUN = cor,
  FUNARGS = list(use = "pairwise.complete.obs"),
  ...
)

## S4 method for signature 'MChromatograms,MChromatograms'
compareChromatograms(
  x,
  y,
  ALIGNFUN = alignRt,
  ALIGNFUNARGS = list(),
  FUN = cor,
  FUNARGS = list(use = "pairwise.complete.obs"),
  ...
)

## S4 method for signature 'MChromatograms'
transformIntensity(object, FUN = identity)
```

Arguments

data	for MChromatograms: a list of Chromatogram() objects.

phenoData	for MChromatograms: either a data.frame, AnnotatedDataFrame describing the phenotypical information of the samples.
featureData	for MChromatograms: either a data.frame or AnnotatedDataFrame with additional information for each row of chromatograms.
...	for MChromatograms: additional parameters to be passed to the matrix constructor, such as nrow, ncol and byrow. For compareChromatograms: ignored.
object	a MChromatograms object.
x	for all methods: a MChromatograms object.
i	for [: numeric, logical or character defining which row(s) to extract.
j	for [: numeric, logical or character defining which columns(s) to extract.
drop	for [: logical(1) whether to drop the dimensionality of the returned object (if possible). The default is drop = FALSE, i.e. each subsetting returns a MChromatograms object (or a Chromatogram object if a single element is extracted).
value	for [<-: the replacement object(s). Can be a list of [Chromatogram()objects or, if length ofiandja object. For `pData<-`: a `data.frame` with the number of rows matching the number of columns of `object`. For `colnames`: a `character` with the new column names.
col	for plot: the color to be used for plotting. Either a vector of length 1 or equal to ncol(x).
lty	for plot: the line type (see plot in the graphics package for more details). Can be either a vector of length 1 or of length equal to ncol(x).
type	for plot: the type of plot (see plot from the graphics package for more details). Can be either a vector of length 1 or of length equal to ncol(x).
xlab	for plot: the x-axis label.
ylab	for plot: the y-axis label.
main	for plot: the plot title. If not provided the mz range will be used as plot title.
name	for $, the name of the pheno data column.
binSize	for bin: numeric(1) with the size of the bins (in seconds).
breaks	For bin: numeric defining the bins. Usually not required as the function calculates the bins automatically based on binSize and the retention time range of chromatograms in the same row.
fun	for bin: function to be used to aggregate the intensity values falling within each bin.
all	for clean: logical(1) whether all 0-intensities should be removed (all = TRUE), or whether 0-intensities adjacent to peaks should be kept (all = FALSE; default).
na.rm	for clean: logical(1) whether all NA intensities should be removed prior to clean 0-intensity data points.

method	character(1). For normalise: defining whether each chromatogram should be normalized to its maximum signal (method = "max") or total signal (method = "sum"). For alignRt: alignment methods (see documentation for alignRt in the Chromatogram() help page. Defaults to method = "closest".
intensity	for filterIntensity: numeric(1) or function to use to filter intensities. See description for details.
y	for alignRt: a Chromatogram() object against which x should be aligned against.
ALIGNFUN	for compareChromatograms: function to align chromatogram x against chromatogram y. Defaults to alignRt.
ALIGNFUNARGS	list of parameters to be passed to ALIGNFUN.
FUN	for transformIntensity: function to transform chromatograms' intensity values. Defaults to FUN = identity.
FUNARGS	for compareChromatograms: list with additional parameters for FUN. Defaults to FUNARGS = list(use = "pairwise.complete.obs").

Details

The MChromatograms class extends the base matrix class and hence allows to store Chromatogram() objects in a two-dimensional array. Each row is supposed to contain Chromatogram objects for one MS data *slice* with a common m/z and rt range. Columns contain Chromatogram objects from the same sample.

Value

For [: the subset of the MChromatograms object. If a single element is extracted (e.g. if i and j are of length 1) a Chromatogram() object is returned. Otherwise (if drop = FALSE, the default, is specified) a MChromatograms object is returned. If drop = TRUE is specified, the method returns a list of Chromatogram objects.

```
For `phenoData`: an `AnnotatedDataFrame` representing the
pheno data of the object.

For `pData`: a `data.frame` representing the pheno data of
the object.

For `$`: the value of the corresponding column in the pheno data
table of the object.

For all other methods see function description.
```

Object creation

MChromatograms are returned by a chromatogram() function from an MSnExp or OnDiskMSnExp. Alternatively, the MChromatograms constructor function can be used.

Data access

- $ and $<-: get or replace individual columns of the object's phenodata.
- colnames and colnames<-: replace or set the column names of the MChromatograms object. Does also set the rownames of the phenoData.
- fData: return the feature data as a data.frame.
- fData<-: replace the object's feature data by passing a data.frame.
- featureData: return the feature data.
- featureData<-: replace the object's feature data.
- featureNames: returns the feature names of the MChromatograms object.
- featureNames<-: set the feature names.
- fvarLabels: return the feature data variable names (i.e. column names).
- isEmpty: returns TRUE if the MChromatograms object or all of its Chromatogram objects is/are empty or contain only NA intensities.
- mz: returns the m/z for each row of the MChromatograms object as a two-column matrix (with columns "mzmin" and "mzmax").
- pData: accesses the phenotypical description of the samples. Returns a data.frame.
- pData<-: replace the phenotype data.
- phenoData: accesses the phenotypical description of the samples. Returns an AnnotatedDataFrame object.
- polarity: returns the polarity of the scans/chromatograms: 1, 0 or -1 for positive, negative or unknown polarity.
- precursorMz: return the precursor m/z from the chromatograms. The method returns a matrix with 2 columns ("mzmin" and "mzmax") and as many rows as there are rows in the MChromatograms object. Each row contains the precursor m/z of the chromatograms in that row. An error is thrown if the chromatograms within one row have different precursor m/z values.
- productMz: return the product m/z from the chromatograms. The method returns a matrix with 2 columns ("mzmin" and "mzmax") and as many rows as there are rows in the MChromatograms object. Each row contains the product m/z of the chromatograms in that row. An error is thrown if the chromatograms within one row have different product m/z values.
- rownames<-: replace the rownames (and featureNames) of the object.

Data subsetting, combining and filtering

- [subset (similar to a matrix) by row and column (with parameters i and j).
- [<- replace individual or multiple elements. value has to be either a single Chromatogram obhect or a list of Chromatogram objects.
- c concatenate (row-wise) MChromatogram objects with the **same number of samples (columns)**.
- filterIntensity: filter each Chromatogram() object within the MChromatograms removing data points with intensities below the user provided threshold. If intensity is a numeric value, the returned chromatogram will only contain data points with intensities > intensity. In addition it is possible to provide a function to perform the filtering. This function is expected to take the input Chromatogram (object) and to return a logical vector with the same length

then there are data points in `object` with `TRUE` for data points that should be kept and `FALSE` for data points that should be removed. See the `filterIntensity` documentation in the `Chromatogram()` help page for details and examples.

Data processing and manipulation

- `alignRt`: align all chromatograms in an `MChromatograms` object against the chromatogram specified with `y`. See documentation on `alignRt` in the `Chromatogram()` help page.
- `bin`: aggregates intensity values of chromatograms in discrete bins along the retention time axis. By default, individual `Chromatogram` objects of one row are binned into the same bins. The function returns a `MChromatograms` object with binned chromatograms.
- `clean`: removes 0-intensity data points. Either all of them (with `all = TRUE`) or all except those adjacent to non-zero intensities (`all = FALSE`; default). See `clean()` documentation for more details and examples.
- `compareChromatograms`: calculates pairwise similarity score between chromatograms in `x` and `y` and returns a similarity matrix with the number of rows corresponding to the number of chromatograms in `x` and the number of columns to the number of chromatograms in `y`. If `y` is missing, a pairwise comparison is performed between all chromatograms in `x`. See documentation on `compareChromatograms` in the `Chromatogram()` help page for details.
- `normalize`, `normalise`: *normalises* the intensities of a chromatogram by dividing them either by the maximum intensity (`method = "max"`) or total intensity (`method = "sum"`) of the chromatogram.
- `transformIntensity`: allows to manipulate the intensity values of all chromatograms using a user provided function. See below for examples.

Data visualization

- `plot`: plots a `MChromatograms` object. For each row in the object one plot is created, i.e. all `Chromatogram()` objects in the same row are added to the same plot. If `nrow(x) > 1` the plot area is split into `nrow(x)` sub-plots and the chromatograms of one row are plotted in each.

Author(s)

Johannes Rainer

See Also

`Chromatogram()]` for the class representing chromatogram data. `[chromatogram()]` for the method to extrac
matogramsobject from aMSnExporOnDiskMSnExp object. `[readSRMData()` for the function to read chromatographic data of an SRM/MRM experiment.

Examples

```
## Creating some chromatogram objects to put them into a MChromatograms object
ints <- abs(rnorm(25, sd = 200))
ch1 <- Chromatogram(rtime = 1:length(ints), ints)
ints <- abs(rnorm(32, sd = 90))
ch2 <- Chromatogram(rtime = 1:length(ints), ints)
ints <- abs(rnorm(19, sd = 120))
```

```
ch3 <- Chromatogram(rtime = 1:length(ints), ints)
ints <- abs(rnorm(21, sd = 40))
ch4 <- Chromatogram(rtime = 1:length(ints), ints)

## Create a MChromatograms object with 2 rows and 2 columns
chrs <- MChromatograms(list(ch1, ch2, ch3, ch4), nrow = 2)
chrs

## Extract the first element from the second column. Extracting a single
## element always returns a Chromatogram object.
chrs[1, 2]

## Extract the second row. Extracting a row or column (i.e. multiple elements
## returns by default a list of Chromatogram objects.
chrs[2, ]

## Extract the second row with drop = FALSE, i.e. return a MChromatograms
## object.
chrs[2, , drop = FALSE]

## Replace the first element.
chrs[1, 1] <- ch3
chrs

## Add a pheno data.
pd <- data.frame(name = c("first sample", "second sample"),
    idx = 1:2)
pData(chrs) <- pd

## Column names correspond to the row names of the pheno data
chrs

## Access a column within the pheno data
chrs$name

## Access the m/z ratio for each row; this will be NA for the present
## object
mz(chrs)


## Data visualization

## Create some random Chromatogram objects
ints <- abs(rnorm(123, mean = 200, sd = 32))
ch1 <- Chromatogram(rtime = seq_along(ints), intensity = ints, mz = 231)
ints <- abs(rnorm(122, mean = 250, sd = 43))
ch2 <- Chromatogram(rtime = seq_along(ints), intensity = ints, mz = 231)
ints <- abs(rnorm(125, mean = 590, sd = 120))
ch3 <- Chromatogram(rtime = seq_along(ints), intensity = ints, mz = 542)
ints <- abs(rnorm(124, mean = 1200, sd = 509))
ch4 <- Chromatogram(rtime = seq_along(ints), intensity = ints, mz = 542)

## Combine into a 2x2 MChromatograms object
```

```
chrs <- MChromatograms(list(ch1, ch2, ch3, ch4), byrow = TRUE, ncol = 2)

## Plot the second row
plot(chrs[2, , drop = FALSE])

## Plot all chromatograms
plot(chrs, col = c("#ff000080", "#00ff0080"))

## log2 transform intensities
res <- transformIntensity(chrs, log2)
plot(res)
```

`meanMzInts` *Combine a list of spectra to a single spectrum*

Description

Combine peaks from several spectra into a single spectrum. Intensity and m/z values from the input spectra are aggregated into a single peak if the difference between their m/z values is smaller than `mzd` or smaller than `ppm` of their m/z. While `mzd` can be used to group mass peaks with a single fixed value, `ppm` allows a m/z dependent mass peak grouping. Intensity values of grouped mass peaks are aggregated with the `intensityFun`, m/z values by the mean, or intensity weighted mean if `weighted = TRUE`.

Usage

```
meanMzInts(
  x,
  ...,
  intensityFun = base::mean,
  weighted = FALSE,
  main = 1L,
  mzd,
  ppm = 0,
  timeDomain = FALSE,
  unionPeaks = TRUE
)
```

Arguments

`x`	`list` of `Spectrum` objects.
`...`	additional parameters that are passed to `intensityFun`.
`intensityFun`	`function` to aggregate the intensity values per m/z group. Should be a function or the name of a function. The function is expected to return a `numeric(1)`.
`weighted`	`logical(1)` whether m/z values per m/z group should be aggregated with an intensity-weighted mean. The default is to report the mean m/z.

main	integer(1) defining the *main* spectrum, i.e. the spectrum which m/z and intensity values get replaced and is returned. By default the *first* spectrum in x is used.
mzd	numeric(1) defining the maximal m/z difference below which mass peaks are considered to represent the same ion/mass peak. Intensity values for such grouped mass peaks are aggregated. If not specified this value is estimated from the distribution of differences of m/z values from the provided spectra (see details).
ppm	numeric(1) allowing to perform a m/z dependent grouping of mass peaks. See details for more information.
timeDomain	logical(1) whether definition of the m/z values to be combined into one m/z is performed on m/z values (timeDomain = FALSE) or on sqrt(mz) (timeDomain = TRUE). Profile data from TOF MS instruments should be aggregated based on the time domain (see details). Note that a pre-defined mzd should also be estimated on the square root of m/z values if timeDomain = TRUE.
unionPeaks	logical(1) whether the union of all peaks (peak groups) from all spectra are reported or only peak groups that contain peaks that are present in the *main* spectrum (defined by main). The default is to report the union of peaks from all spectra.

Details

For general merging of spectra, the mzd and/or ppm should be manually specified based on the precision of the MS instrument. Peaks from spectra with a difference in their m/z being smaller than mzd or smaller than ppm of their m/z are grouped into the same final peak.

Some details for the combination of consecutive spectra of an LCMS run:

The m/z values of the same ion in consecutive scans (spectra) of a LCMS run will not be identical. Assuming that this random variation is much smaller than the resolution of the MS instrument (i.e. the difference between m/z values within each single spectrum), m/z value groups are defined across the spectra and those containing m/z values of the main spectrum are retained. The maximum allowed difference between m/z values for the same ion is estimated as in estimateMzScattering(). Alternatively it is possible to define this maximal m/z difference with the mzd parameter. All m/z values with a difference smaller than this value are combined to a m/z group. Intensities and m/z values falling within each of these m/z groups are aggregated using the intensity_fun and mz_fun, respectively. It is highly likely that all QTOF profile data is collected with a timing circuit that collects data points with regular intervals of time that are then later converted into m/z values based on the relationship t = k * sqrt(m/z). The m/z scale is thus non-linear and the m/z scattering (which is in fact caused by small variations in the time circuit) will thus be different in the lower and upper m/z scale. m/z-intensity pairs from consecutive scans to be combined are therefore defined by default on the square root of the m/z values. With timeDomain = FALSE, the actual m/z values will be used.

Value

Spectrum with m/z and intensity values representing the aggregated values across the provided spectra. The returned spectrum contains the union of all peaks from all spectra (if unionPeaks = TRUE), or the same number of m/z and intensity pairs than the spectrum with index main in x (if unionPeaks = FALSE. All other spectrum data (such as retention time etc) is taken from the *main* spectrum.

Note

This allows e.g. to combine profile-mode spectra of consecutive scans into the values for the *main* spectrum. This can improve centroiding of profile-mode data by increasing the signal-to-noise ratio and is used in the `combineSpectraMovingWindow()` function.

Author(s)

Johannes Rainer, Sigurdur Smarason

See Also

`estimateMzScattering()` for a function to estimate m/z scattering in consecutive scans.

`estimateMzResolution()` for a function estimating the m/z resolution of a spectrum.

`combineSpectraMovingWindow()` for the function to combine consecutive spectra of an `MSnExp` object using a moving window approach.

Other spectra combination functions: `consensusSpectrum()`

Examples

```
library(MSnbase)
## Create 3 example profile-mode spectra with a resolution of 0.1 and small
## random variations on these m/z values on consecutive scans.
set.seed(123)
mzs <- seq(1, 20, 0.1)
ints1 <- abs(rnorm(length(mzs), 10))
ints1[11:20] <- c(15, 30, 90, 200, 500, 300, 100, 70, 40, 20) # add peak
ints2 <- abs(rnorm(length(mzs), 10))
ints2[11:20] <- c(15, 30, 60, 120, 300, 200, 90, 60, 30, 23)
ints3 <- abs(rnorm(length(mzs), 10))
ints3[11:20] <- c(13, 20, 50, 100, 200, 100, 80, 40, 30, 20)

## Create the spectra.
sp1 <- new("Spectrum1", mz = mzs + rnorm(length(mzs), sd = 0.01),
    intensity = ints1)
sp2 <- new("Spectrum1", mz = mzs + rnorm(length(mzs), sd = 0.01),
    intensity = ints2)
sp3 <- new("Spectrum1", mz = mzs + rnorm(length(mzs), sd = 0.009),
    intensity = ints3)

## Combine the spectra
sp_agg <- meanMzInts(list(sp1, sp2, sp3))

## Plot the spectra before and after combining
par(mfrow = c(2, 1), mar = c(4.3, 4, 1, 1))
plot(mz(sp1), intensity(sp1), xlim = range(mzs[5:25]), type = "h", col = "red")
points(mz(sp2), intensity(sp2), type = "h", col = "green")
points(mz(sp3), intensity(sp3), type = "h", col = "blue")
plot(mz(sp_agg), intensity(sp_agg), xlim = range(mzs[5:25]), type = "h",
    col = "black")
```

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

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 visualisation 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 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 or an instance of class OnDiskMSnExp. If the latter contains data from multiple files, a warning will be issued and the first one 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. Ignored when object is an OnDiskMSnExp.

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

Examples

```
## Not run:
    ## downloads the data
    library("rpx")
    px1 <- PXDataset("PXD000001")
    (i <- grep("TMT.+mzML", pxfiles(px1), value = TRUE))
    mzf <- pxget(px1, i)

    ## Using an mzRpwiz object
    ## 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)

## Using an OnDiskMSnExp object and accessors
msn <- readMSData(mzf, mode = "onDisk")

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

## the map
M3 <- MSmap(msn, ms1[rtsel], 521, 523, .005)
plot(M3, aspect = 1, allTicks = FALSE)

## With some MS2 spectra
i <- ms1[which(rtsel)][1]
j <- ms1[which(rtsel)][2]
M4 <- MSmap(msn, i:j, 100, 1000, 1)
plot3D(M4)

## End(Not run)
```

`MSnbaseOptions` *MSnbase options*

Description

MSnbase defined a few options globally using the standard R options mechanism. The current values of these options can be queried with `MSnbaseOptions`. The options are:

- `verbose`: defines a session-wide verbosity flag, that is used if the `verbose` argument in individual functions is not set.
- `PARALLEL_THRESH`: defines the minimum number of spectra per file necessary before using parallel processing.
- `fastLoad`: `logical(1)`. If `TRUE` performs faster data loading for all methods of OnDiskMSnExp that load data from the original files (such as `spectrapply()`). Users experiencing data I/O errors (observed mostly on macOS systems) should set this option to `FALSE`.

Usage

```
MSnbaseOptions()

isMSnbaseVerbose()

setMSnbaseVerbose(opt)
```

```
setMSnbaseParallelThresh(opt = 1000)

setMSnbaseFastLoad(opt = TRUE)

isMSnbaseFastLoad()
```

Arguments

opt	The value of the new option

Details

isMSnbaseVerbose is one wrapper for the verbosity flag, also available through options("MSnbase")$verbose.

There are also setters to set options individually. When run without argument, the verbosity setter inverts the current value of the option.

Value

A list of MSnbase options and the single option values for the individual accessors.

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.

In version 1.19.12, the polarity slot had been added to the "Spectrum" class (previously in "Spectrum1"). Hence, "MSnExp" objects created prior to this change will not be valid anymore, since all MS2 spectra will be missing the polarity slot. Object can be appropriately updated using the updateObject method.

The feature variables in the feature data slot will depend on the file. See also the documentation in the mzR package that parses the raw data files and produces these data.

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(x = "Spectrum", y = "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. Parameter `msLevel.` allows to restrict peak picking to spectra of certain MS level(s). See `pickPeaks` documentation for more details and examples.

estimateNoise `signature(object = "MSnExp")`: Estimates the noise in all profile spectra of `object`. See `estimateNoise` documentation for more details and examples.

plot `signature(x = "MSnExp", y = "missing")`: Plots 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. Also for `OnDiskMSnExp` objects.

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 (or feature belong to protein groups) with more than one member. The latter is defined by extracting a feature variable (default is `"nprot"`). Also removes non-identified features.

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.

isolationWindow `signature(object = "MSnExp", ...)`: Returns the isolation window offsets for the MS2 spectra. See `isolationWindow` in the `mzR` package for details.

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

`isCentroided(object, k = 0.025, qtl = 0.9, verbose = TRUE)` A heuristic assessing if the spectra in the `object` are in profile or centroided mode. The function takes the `qtl`th quantile top peaks, then calculates the difference between adjacent M/Z value and returns `TRUE` if the first quartile is greater than `k`. (See `MSnbase:::.isCentroided` for the code.) If `verbose` (default), a table indicating mode for all MS levels is printed.

The function has been tuned to work for MS1 and MS2 spectra and data centroided using different peak picking algorithms, but false positives can occur. See https://github.com/lgatto/MSnbase/issues/131 for details. For whole experiments, where all MS1 and MS2 spectra are expected to be in the same, albeit possibly different modes, it is advised to assign the majority result for MS1 and MS2 spectra, rather than results for individual spectra. See an example below.

as `signature(object = "MSnExp", "data.frame")`: Coerces the `MSnExp` object to a four-column `data.frame` with columns `"file"` (file index in `object`), `"rt"` (retention time), `"mz"` (m/z values) and `"i"` (intensity values).

as `signature(object = "MSnExp", "MSpectra")`: Coerces the `MSnExp` object to a `MSpectra` object with all feature annotations added as metadata columns (`mcols`).

Clarifications regarding scan/acquisition numbers and indices:

A `spectrumId` (or `spectrumID`) is a vendor specific field in the mzML file that contains some information about the run/spectrum, e.g.: `controllerType=0 controllerNumber=1 scan=5281 file=2`.

`acquisitionNum` is a more a less sanitize spectrum id generated from the `spectrumId` field by `mzR` (see https://github.com/sneumann/mzR/blob/master/src/pwiz/data/msdata/MSData.cpp#L552-L580).

`scanIndex` is the `mzR` generated sequence number of the spectrum in the raw file (which doesn't have to be the same as the `acquisitionNum`).

See also this issue: https://github.com/lgatto/MSnbase/issues/525.

Filtering and subsetting functions:

filterRt `signature(object = "MSnExp", rt = "numeric", msLevel. = "numeric")`: Retains MS spectra of level `msLevel.` with a retention times within `rt[1]` and `rt[2]`.

filterMsLevel `signature(object = "MSnExp", msLevel. = "numeric")`: Retains MS spectra of level `msLevel.`.

filterPolarity `signature(object = "MSnExp", polarity. = "numeric")`: Retains MS spectra of polarity `polarity.`.

filterMz `signature(object = "MSnExp", mz = "numeric", msLevel. = "numeric")`. See `filterMz` for details.

filterFile `signature(object = "MSnExp", file)`: Retains MS data of files matching the file index or file name provided with parameter `file`.

filterAcquisitionNum

filterEmptySpectra `signature(object = "MSnExp")`: Remove empty spectra from `object` (see `isEmpty`).

filterPrecursorScan `signature(object = "MSnExp", acquisitionNum = "numeric")`: Retain parent (e.g. MS1) and children scans (e.g. MS2) of `acquisitionNum`. See `OnDiskMSnExp` for an example.

splitByFile `signature(object = "MSnExp", f = "factor")`: split a `MSnExp` object by file into a `list` of `MSnExp` objects given the grouping in `factor f`.

filterPrecursorMz `signature(object = "MSnExp", mz, ppm = 10)`: retain spectra with a precursor m/z equal or similar to the one defined with parameter `mz`. Parameter `ppm` allows to define an accepted difference between the provided m/z and the spectrum's m/z.

filterIsolationWindow `signature(object = "MSnExp", mz)`: retain spectra with isolation windows that contain (which m/z range contain) the specified m/z.

Author(s)

Laurent Gatto

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.

The "OnDiskMSnExp" manual page contains further details and examples.

chromatogram to extract chromatographic data from a MSnExp or OnDiskMSnExp object.

write for the function to write the data to mzML or mzXML file(s).

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

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 class 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. Proteomics expression data available as spreadsheets, as produced by third-party software such as Proteome Discoverer, MaxQuant, ... can be imported using the readMSnSet and readMSnSet2 functions.

Coercion methods are also available to transform MSnSet objects to IBSpectra, to data.frame and to/from ExpressionSet and SummarizedExperiment objects. In the latter case, the metadata available in the protocolData, experimentData are completely dropped, and only the logging information of the processingData slot is retained. All these metadata can be subsequently be added using the addMSnSetMetadata (see examples below). When converting a SummarizedExperiment to an MSnSet, the respective metadata slots will be populated if available in the SummarizedExperiment metadata.

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. This slot can be accessed as a data.frame with pData and be replaced by a new valid (i.e. of compatible dimensions and row names) data.frame with pData()<-.

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. This slot can be accessed as a data.frame with fData and be replaced by a new valid (i.e. of compatible dimensions and row names) data.frame with fData()<-.

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.

acquisitionNum `acquisitionNum(signature(object = "MSnSet"))`: Returns the a numeric vector with acquisition number of each spectrum. The vector names are the corresponding spectrum names. The information is extracted from the object's `featureData` slot.

fromFile `fromFile(signature(object = "MSnSet"))`: get the index of the file (in `fileNames(object)`) from which the raw spectra from which the corresponding feature were originally read. The relevant information is extracted from the object's `featureData` slot.

Returns a numeric vector with names corresponding to the spectrum names.

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(,"IBSpectra") `signature(x = "MSnSet")`: Coerce object from `MSnSet` to `IBSpectra` from the `isobar` package.

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

as(,"SummarizedExperiment") `signature(x = "MSnSet")`: Coerce object from `MSnSet` to `SummarizedExperiment`. Only part of the metadata is retained. See `addMSnSetMetadata` and the example below for details.

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, ..., verbose = isMSnbaseVerbose())`: 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 summerised 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. (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.

filterMsLevel signature(object = "MSnSet", msLevel. = "numeric", fcol = "character") Keeps only spectra with level msLevel., as defined by the fcol feature variable (default is "msLevel").

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.

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

trimws signature(object = "MSnSet", ...)Trim leading and/or trailing white spaces in the feature data slot. Also available for data.frame objects. See ?base::trimws for 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 = "character", high = "character", fnames = "logical", nmax = "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 (or feature belong to protein groups) with more than one member. The latter is defined by extracting a feature variable (default is `"nprot"`). Also removes non-identified features.

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.

addMSnSetMetadata `signature(x, y)` When coercing an `MSnSet` `y` to a `SummarizedExperiment` `x` with `x <- as(y, "SummarizedExperiment")`, most of `y`'s metadata is lost. Only the file names, the processing log and the MSnbase version from the `processingData` slots are passed along. The `addMSnSetMetadata` function can be used to add the complete `processingData`, `experimentData` and `protocolData` slots. The downside of this is that MSnbase is now required to use the `SummarizedExperiment` object.

Author(s)

Laurent Gatto

See Also

"eSet", "ExpressionSet" and quantify. MSnSet quantitation values and annotation can be exported to a file with write.exprs. See readMSnSet to create and MSnSet using data available in a spreadsheet or data.frame.

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")


## Coercion
if (require("SummarizedExperiment")) {
    data(msnset)
    se <- as(msnset, "SummarizedExperiment")
    metadata(se) ## only logging
    se <- addMSnSetMetadata(se, msnset)
    metadata(se) ## all metadata
    msnset2 <- as(se, "MSnSet")
    processingData(msnset2)
}

as(msnset, "ExpressionSet")
```

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, log, featureDAta) The class constructor that takes a list of valid MSnSet instances as input x, an optional logging list, and an optional feature metadata data.frame.

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().

featureData: Object of class DataFrame that stores metadata for each object in the x slot. The number of rows of this data.frame must be equal to the number of items in the x slot and their respective (row)names must be identical.

.__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. The replacement method is also available.

show Display the object by printing a short summary.

lapply(x, FUN, ...) Apply function FUN to each element of the input x. If the application of FUN returns and MSnSet, then the return value is an MSnSetList, otherwise a list.

`sapply(x, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)` A `lapply` wrapper that simplifies the ouptut to a vector, matric or array is possible. See ?`base::sapply` for details. .

`fData` Returns the features metadata `featureData` slot.

`fData<-` Features metadata `featureData` replacement method.

Author(s)

Laurent Gatto

See Also

The `commonFeatureNames` function to select common features among `MSnSet` instances.

Examples

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

## The MSnSetList class
##  for an unnamed list, names are set to indices
msnl <- MSnSetList(list(tan2009r1, tan2009r2))
names(msnl)
##  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, method = "quantiles") ## an MSnSetList

fData(msnl)
fData(msnl)$X <- sapply(msnl, nrow)
fData(msnl)

## 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))
```

MSpectra *List of Spectrum objects along with annotations*

Description

MSpectra (Mass Spectra) objects allow to collect one or more Spectrum object(s) (Spectrum1 or Spectrum2) in a list-like structure with the possibility to add arbitrary annotations to each individual Spectrum object. These can be accessed/set with the mcols() method.

MSpectra objects can be created with the MSpectra function.

Functions to access the individual spectra's attributes are available (listed below).

writeMgfData exports a MSpectra object to a file in MGF format. All metadata columns present in mcols are exported as additional fields with the capitalized column names used as field names (see examples below).

Usage

```
MSpectra(..., elementMetadata = NULL)

## S4 method for signature 'MSpectra'
mz(object)

## S4 method for signature 'MSpectra'
intensity(object)

## S4 method for signature 'MSpectra'
rtime(object)

## S4 method for signature 'MSpectra'
precursorMz(object)

## S4 method for signature 'MSpectra'
precursorCharge(object)

## S4 method for signature 'MSpectra'
precScanNum(object)

## S4 method for signature 'MSpectra'
precursorIntensity(object)

## S4 method for signature 'MSpectra'
acquisitionNum(object)
```

```
## S4 method for signature 'MSpectra'
scanIndex(object)

## S4 method for signature 'MSpectra,ANY'
peaksCount(object)

## S4 method for signature 'MSpectra'
msLevel(object)

## S4 method for signature 'MSpectra'
tic(object)

## S4 method for signature 'MSpectra'
ionCount(object)

## S4 method for signature 'MSpectra'
collisionEnergy(object)

## S4 method for signature 'MSpectra'
fromFile(object)

## S4 method for signature 'MSpectra'
polarity(object)

## S4 method for signature 'MSpectra'
smoothed(object)

## S4 method for signature 'MSpectra'
isEmpty(x)

## S4 method for signature 'MSpectra'
centroided(object)

## S4 method for signature 'MSpectra'
isCentroided(object)

## S4 method for signature 'MSpectra'
writeMgfData(object, con = "spectra.mgf", COM = NULL, TITLE = NULL)

## S4 method for signature 'MSpectra'
clean(object, all = FALSE, msLevel. = msLevel., ...)

## S4 method for signature 'MSpectra'
removePeaks(object, t, msLevel., ...)

## S4 method for signature 'MSpectra'
filterMz(object, mz, msLevel., ...)
```

```
## S4 method for signature 'MSpectra'
pickPeaks(
  object,
  halfWindowSize = 3L,
  method = c("MAD", "SuperSmoother"),
  SNR = 0L,
  refineMz = c("none", "kNeighbors", "kNeighbours", "descendPeak"),
  msLevel. = unique(msLevel(object)),
  ...
)

## S4 method for signature 'MSpectra'
smooth(
  x,
  method = c("SavitzkyGolay", "MovingAverage"),
  halfWindowSize = 2L,
  ...
)

## S4 method for signature 'MSpectra'
filterMsLevel(object, msLevel.)
```

Arguments

`...`	For `MSpectra`: Spectrum object(s) or a `list` of Spectrum objects. For all other methods optional arguments passed along.
`elementMetadata`	For `MSpectra`: DataFrame with optional information that should be added as metadata information (`mcols`) to the object. The number of rows has to match the number of Spectrum objects, each row is expected to represent additional metadata information for one spectrum.
`object`	For all functions: a `MSpectra` object.
`x`	For all functions: a `MSpectra` object.
`con`	For `writeMgfData`: `character(1)` defining the file name of the MGF file.
`COM`	For `writeMgfData`: optional `character(1)` providing a comment to be added to the file.
`TITLE`	For `writeMgfData`: optional `character(1)` defining the title for the MGF file.
`all`	For `clean`: if `FALSE` original 0-intensity values are retained around peaks.
`msLevel.`	For `clean`, `removePeaks`, `filterMz`, `pickPeaks`: optionally specify the MS level(s) of the spectra on which the operation should be performed. For `filterMsLevels`: MS level(s) to which the `MSpectra` should be reduced.
`t`	For `removePeaks`: `numeric(1)` specifying the threshold below which intensities are set to 0.
`mz`	For `filterMz`: `numeric(2)` defining the lower and upper m/z for the filter. See `filterMz()` for details.

halfWindowSize	For pickPeaks and smooth: see pickPeaks() and smooth() for details.
method	For pickPeaks and smooth: see pickPeaks() and smooth() for details.
SNR	For pickPeaks: see pickPeaks() for details.
refineMz	For pickPeaks: see pickPeaks() for details.

Details

MSpectra inherits all methods from the SimpleList class of the S4Vectors package. This includes lapply and other data manipulation and subsetting operations.

Constructor

New MSpectra can be created with the MSpectra(...) function where ... can either be a single Spectrum object or a list of Spectrum objects (Spectrum1 and/or Spectrum2).

Accessing spectrum attributes

These methods allow to access the attributes and values of the individual Spectrum (Spectrum1 or Spectrum2) objects within the list.

- mz return the m/z values of each spectrum as a list of numeric vectors.
- intensity return the intensity values of each spectrum as a list of numeric vectors.
- rtime return the retention time of each spectrum as a numeric vector with length equal to the length of object.
- precursorMz, precursorCharge, precursorIntensity, precScanNum return precursor m/z values, charge, intensity and scan number for each spectrum as a numeric (or integer) vector with length equal to the length of object. Note that for Spectrum1 objects NA will be returned.
- acquisitionNum and scanIndex return the acquisition number of each spectrum and its scan index as an integer vector with the same length than object.
- ionCount and tic return the ion count and total ion current of each spectrum.
- peaksCount returns the number of peaks for each spectrum as a integer vector.
- msLevel returns the MS level of each spectrum.
- collisionEnergy returns the collision energy for each spectrum or NA for Spectrum1 objects.
- polarity returns the spectra's polarity.
- fromFile returns the index from the (e.g. mzML) file the spectra where from. This applies only for spectra read using the readMSData() function.
- smoothed whether spectra have been smoothed (i.e. processed with the smooth() method. Returns a logical of length equal to the number of spectra.
- isEmpty returns TRUE for spectra without peak data.
- centroided, isCentroided returns for each spectrum whether it contains *centroided* data. While centroided returns the internal attribute of each spectrum, isCentroided tries to guess whether spectra are centroided from the actual peak data.

Data manipulation methods

- `clean` *cleans* each spectrum. See `clean()` for more details.
- `pickPeaks` performs peak picking to generate centroided spectra. See `pickPeaks()` for more details.
- `removePeaks` removes peaks lower than a threshold `t`. See `removePeaks()` for more details.
- `smooth` *smooths* spectra. See `smooth()` for more details.

Filtering and subsetting

- `[` can be used to subset the `MSpectra` object.
- `filterMsLevel` filters `MSpectra` to retain only spectra from certain MS level(s).
- `filterMz` filters the spectra by the specified `mz` range. See `filterMz()` for details.

Note

Note that the Spectra package provides a more robust and efficient infrastructure for mass spectrometry data handling and analysis. So, wherever possible, the newer *Spectra* package should be used instead of the *MSnbase*.

For backward compatibility, it is however possible to convert between the `MSpectra` and the newer `Spectra` objects:

- A `Spectra` object can be coerced to a `MSpectra` using `as(sps, "MSpectra")` where `sps` is a `Spectra` object.
- The `extractSpectraData()` function can be used to extract the data from a `MSpectra` as a `DataFrame`, which can then be used to create a `Spectra` object.

Author(s)

Johannes Rainer

Examples

```
## Create from Spectrum objects
sp1 <- new("Spectrum1", mz = c(1, 2, 4), intensity = c(4, 5, 2))
sp2 <- new("Spectrum2", mz = c(1, 2, 3, 4), intensity = c(5, 3, 2, 5),
    precursorMz = 2)

spl <- MSpectra(sp1, sp2)
spl
spl[[1]]

## Add also metadata columns
mcols(spl)$id <- c("a", "b")
mcols(spl)

## Create a MSpectra with metadata
spl <- MSpectra(sp1, sp2, elementMetadata = DataFrame(id = c("a", "b")))
```

```
mcols(spl)
mcols(spl)$id

## Extract the mz values for the individual spectra
mz(spl)

## Extract the intensity values for the individual spectra
intensity(spl)

## Extract the retention time values for the individual spectra
rtime(spl)

## Extract the precursor m/z of each spectrum.
precursorMz(spl)

## Extract the precursor charge of each spectrum.
precursorCharge(spl)

## Extract the precursor scan number for each spectrum.
precScanNum(spl)

## Extract the precursor intensity of each spectrum.
precursorIntensity(spl)

## Extract the acquisition number of each spectrum.
acquisitionNum(spl)

## Extract the scan index of each spectrum.
scanIndex(spl)

## Get the number of peaks per spectrum.
peaksCount(spl)

## Get the MS level of each spectrum.
msLevel(spl)

## Get the total ion current for each spectrum.
tic(spl)

## Get the total ion current for each spectrum.
ionCount(spl)

## Extract the collision energy for each spectrum.
collisionEnergy(spl)

## Extract the file index for each spectrum.
fromFile(spl)

## Get the polarity for each spectrum.
polarity(spl)

## Whether spectra are smoothed (i.e. processed with the `smooth`
## function).
```

```
smoothed(spl)

## Are spectra empty (i.e. contain no peak data)?
isEmpty(spl)

## Do the spectra contain centroided data?
centroided(spl)

## Do the spectra contain centroided data? Whether spectra are centroided
## is estimated from the peak data.
isCentroided(spl)

## Export the spectrum list to a MGF file. Values in metadata columns are
## exported as additional field for each spectrum.
tmpf <- tempfile()
writeMgfData(spl, tmpf)

## Evaluate the written output. The ID of each spectrum (defined in the
## "id" metadata column) is exported as field "ID".
readLines(tmpf)

## Set mcols to NULL to avoid export of additional data fields.
mcols(spl) <- NULL
file.remove(tmpf)

writeMgfData(spl, tmpf)
readLines(tmpf)

## Filter the object by MS level
filterMsLevel(spl, msLevel. = 1)
```

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. The column names are kept as they are defined in the original files and are thus not all going to be valid colnames. To access them using the dollar operator, use backticks. 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.

`MoleculeFeatures`: Object of class `"data.frame"` storing the molecule features.

`MoleculeEvidence`: Object of class `"data.frame"` storing the molecule evidence.

`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 (SML) `data.frame`.

moleculeFeatures `signature(object = "MzTab")`: returns the small molecules features (SMF) `data.frame`.

moleculeEvidence `signature(object = "MzTab")`: returns the small molecule identification evidence (SME) `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) and Steffen Neuman (see https://github.com/lgatto/MSnbase/pull/500).

References

The mzTab format is a light-weight, tab-delimited file format for proteomics data. Version mzTab 1.0 is aimed at proteomics, mzTab-M 2.0 was adapted to metabolomics. See https://github.com/HUPO-PSI/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.

Hoffmann N, Rein J, Sachsenberg T, et al. mzTab-M: A Data Standard for Sharing Quantitative Results in Mass Spectrometry Metabolomics. Anal Chem. 2019;91(5):3302-3310. doi:10.1021/acs.analchem.8b04310 PubMed PMID: 30688441; PubMed Central PMCID: PMC6660005.

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://raw.githubusercontent.com/HUPO-PSI/mzTab/master/examples/1_0-Proteomics-Release/"

## 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]
```

`naplot` *Overview of missing value*

Description

Visualise missing values as a heatmap and barplots along the samples and features.

Usage

```
naplot(
  object,
  verbose = isMSnbaseVerbose(),
  reorderRows = TRUE,
  reorderColumns = TRUE,
  ...
)
```

Arguments

object	An object of class MSnSet.
verbose	If verbose (default is isMSnbaseVerbose()), print a table of missing values.
reorderRows	If reorderRows (default is TRUE) rows are ordered by number of NA.
reorderColumns	If reorderColumns (default is TRUE) columns are ordered by number of NA.
...	Additional parameters passed to image2.

Value

Used for its side effect. Invisibly returns NULL

Author(s)

Laurent Gatto

Examples

```
data(naset)
naplot(naset)
```

---

nFeatures *How many features in a group?*

Description

This function computes the number of features in the group defined by the feature variable fcol and appends this information in the feature data of object.

Usage

```
nFeatures(object, fcol)
```

Arguments

object	An instance of class MSnSet.
fcol	Feature variable defining the feature grouping structure.

Value

An updated MSnSet with a new feature variable fcol.nFeatures.

Author(s)

Laurent Gatto

Examples

```
library(pRolocdata)
data("hyperLOPIT2015ms3r1psm")
hyperLOPIT2015ms3r1psm <- nFeatures(hyperLOPIT2015ms3r1psm,
                                    "Protein.Group.Accessions")
i <- c("Protein.Group.Accessions", "Protein.Group.Accessions.nFeatures")
fData(hyperLOPIT2015ms3r1psm)[1:10, i]
```

---

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, .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. diff.median translates all samples (columns) so that they all match the grand 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
```

`normToReference` *Combine peptides into proteins.*

Description

This function combines peptides into their proteins by normalising the intensity values to a reference run/sample for each protein.

Usage

```
normToReference(
  x,
  group,
  reference = .referenceFractionValues(x = x, group = group)
)
```

Arguments

`x`	`matrix`, `exprs` matrix of an MSnSet object.
`group`	`double` or `factor`, grouping variable, i.e. protein accession; has to be of length equal `nrow(x)`.
`reference`	`double`, vector of reference values, has to be of the same length as `group` and `nrow(x)`.

Details

This function is not intented to be used directly (that's why it is not exported via NAMESPACE). Instead the user should use combineFeatures.

The algorithm is described in Nikolovski et al., briefly it works as follows:

1. Find reference run (column) for each protein (grouped rows). We use the run (column) with the lowest number of NA. If multiple candidates are available we use the one with the highest intensity. This step is skipped if the user use his own reference vector.
2. For each protein (grouped rows) and each run (column):
   (a) Find peptides (grouped rows) shared by the current run (column) and the reference run (column).
   (b) Sum the shared peptides (grouped rows) for the current run (column) and the reference run (column).
   (c) The ratio of the shared peptides (grouped rows) of the current run (column) and the reference run (column) is the new intensity for the current protein for the current run.

Value

a matrix with one row per protein.

Author(s)

Sebastian Gibb mail@sebastiangibb.de, Pavel Shliaha

References

Nikolovski N, Shliaha PV, Gatto L, Dupree P, Lilley KS. Label-free protein quantification for plant Golgi protein localization and abundance. Plant Physiol. 2014 Oct;166(2):1033-43. DOI: 10.1104/pp.114.245589. PubMed PMID: 25122472.

See Also

combineFeatures

Examples

```
library("MSnbase")
data(msnset)

# choose the reference run automatically
combineFeatures(msnset, groupBy=fData(msnset)$ProteinAccession)

# use a user-given reference
combineFeatures(msnset, groupBy=fData(msnset)$ProteinAccession,
 reference=rep(2, 55))
```

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.

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

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.

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(x, groupBy)
```

Arguments

x	An instance of class "MSnSet".
groupBy	An object of class factor defining how to summerise the features. (Note that this parameter was previously named fcol and referred to a feature variable label. This has been updated in version 1.19.12 for consistency with other functions.)

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 features 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(groupBy)) by ncol(x)

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

Author(s)

Laurent Gatto and Sebastian Gibb

Examples

```
data(msnset)
n <- 2
msnset <- topN(msnset, groupBy = fData(msnset)$ProteinAccession, n)
m <- nQuants(msnset, groupBy = fData(msnset)$ProteinAccession)
msnset2 <- combineFeatures(msnset,
                           groupBy = fData(msnset)$ProteinAccession,
                           method = sum)
```

```
stopifnot(dim(n) == dim(msnset2))
head(exprs(msnset2))
head(exprs(msnset2) * (n/m))
```

OnDiskMSnExp-class *The* OnDiskMSnExp *Class for MS Data And Meta-Data*

Description

Like the MSnExp class, the OnDiskMSnExp class encapsulates data and meta-data for mass spectrometry experiments, but does, in contrast to the former, not keep the spectrum data in memory, but fetches the M/Z and intensity values on demand from the raw files. This results in some instances to a reduced performance, has however the advantage of a much smaller memory footprint.

Details

The OnDiskMSnExp object stores many spectrum related information into the featureData, thus, some calls, like rtime to retrieve the retention time of the individual scans does not require the raw data to be read. Only M/Z and intensity values are loaded on-the-fly from the original files. Extraction of values for individual scans is, for mzML files, very fast. Extraction of the full data (all spectra) are performed in a per-file parallel processing strategy.

Data manipulations related to spectras' M/Z or intensity values (e.g. removePeaks or clean) are (for OnDiskMSnExp objects) not applied immediately, but are stored for later execution into the spectraProcessingQueue. The manipulations are performed *on-the-fly* upon data retrieval. Other manipulations, like removal of individual spectra are applied directly, since the corresponding data is available in the object's featureData slot.

Objects from the Class

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

Slots

backend**:** Character string specifying the used backend.

spectraProcessingQueue**:** list of ProcessingStep objects defining the functions to be applied *on-the-fly* to the spectra data (M/Z and intensity duplets).

assayData**:** Object of class "environment" that is however empty, as no spectrum data is stored. 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). 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 "MSnExp", directly. Class "pSet", by class "MSnExp", distance 3. Class "VersionedBiobase", by class "pSet", distance 4. Class "Versioned", by class "pSet", distance 5.

Getter/setter methods

(in alphabetical order) See also methods for MSnExp or pSet objects.

[`object[i]`:subset the `OnDiskMSnExp` by spectra. `i` can be a `numeric` or `logical` vector specifying to which spectra the data set should be reduced (with `i` being the index of the spectrum in the object's `featureData`).

The method returns a `OnDiskMSnExp` object with the data sub-set.

[[`object[[i]]`: extract s single spectrum from the `OnDiskMSnExp` object `object`. Argument `i` can be either numeric or character specifying the index or the name of the spectrum in the object (i.e. in the `featureData`). The relevant information will be extracted from the corresponding raw data file.

The method returns a `Spectrum1` object.

acquisitionNum `acquisitionNum(signature(object="OnDiskMSnExp"))`: get the acquisition number of each spectrum in each individual file. The relevant information is extracted from the object's `featureData` slot.

Returns a numeric vector with names corresponding to the spectrum names.

assayData `assayData(signature(object = "OnDiskMSnExp"))`: Extract the full data, i.e. read all spectra from the original files, apply all processing steps from the `spectraProcessingQueue` slot and return the data. Due to the required processing time accessing the full data should be avoided wherever possible.

Returns an `environment`.

centroided,centroided<- `centroided(signature(object="OnDiskMSnExp", msLevel, = "numeric"))`: whether individual spectra are centroided or uncentroided. The relevant information is extracted from the object's `featureData` slot. Returns a logical vector with names corresponding to the spectrum names. Use `centroided(object) <- value` to update the information, with value being a logical vector of length equal to the number of spectra in the experiment.

`isCentroided(object, k = 0.025, qtl = 0.9, verbose = TRUE)` A heuristic assessing if the spectra in the `object` are in profile or centroided mode. The function takes the `qtl`th quantile top peaks, then calculates the difference between adjacent M/Z value and returns `TRUE` if the first

quartile is greater than k. (See MSnbase:::.isCentroided for the code.) If verbose (default), a table indicating mode for all MS levels is printed.

The function has been tuned to work for MS1 and MS2 spectra and data centroided using different peak picking algorithms, but false positives can occur. See https://github.com/lgatto/MSnbase/issues/131 for details. For whole experiments, where all MS1 and MS2 spectra are expected to be in the same, albeit possibly different modes, it is advised to assign the majority result for MS1 and MS2 spectra, rather than results for individual spectra.

See also isCentroidedFromFile that accessed the mode directly from the raw data file.

fromFile fromFile(signature(object = "OnDiskMSnExp")): get the index of the file (in fileNames(object)) from which the spectra were read. The relevant information is extracted from the object's featureData slot.

Returns a numeric vector with names corresponding to the spectrum names.

intensity intensity(signature(object="OnDiskMSnExp")): return the intensities from each spectrum in the data set. Intensities are first read from the raw files followed by an optional processing (depending on the processing steps defined in the spectraProcessingQueue). To reduce the amount of required memory, this is performed on a per-file basis. The BPPARAM argument allows to specify how and if parallel processing should be used. Information from individual files will be processed in parallel (one process per original file).

The method returns a list of numeric intensity values. Each list element represents the intensities from one spectrum.

ionCount ionCount(signature(object="OnDiskMSnExp",BPPARAM=bpparam())): extract the ion count (i.e. sum of intensity values) for each spectrum in the data set. The relevant data has to be extracted from the raw files (with eventually applying processing steps). The BPPARAM argument can be used to define how and if parallel processing should be used. Information from individual files will be processed in parallel (one process per original file).

Returns a numeric vector with names corresponding to the spectrum names.

isolationWindowLowerMz isolationWindowLowerMz(object = "OnDiskMSnExp"): return the lower m/z boundary for the isolation window.

Returns a numeric vector of length equal to the number of spectra with the lower m/z value of the isolation window or NA if not specified in the original file.

isolationWindowUpperMz isolationWindowUpperMz(object = "OnDiskMSnExp"): return the upper m/z boundary for the isolation window.

Returns a numeric vector of length equal to the number of spectra with the upper m/z value of the isolation window or NA if not specified in the original file.

length length(signature(object="OnDiskMSnExp")): Returns the number of spectra of the current experiment.

msLevel msLevel(signature(object = "OnDiskMSnExp")): extract the MS level from the spectra. The relevant information is extracted from the object's featureData slot.

Returns a numeric vector with names corresponding to the spectrum names.

mz mz(signature(object="OnDiskMSnExp")): return the M/Z values from each spectrum in the data set. M/Z values are first read from the raw files followed by an optional processing (depending on the processing steps defined in the spectraProcessingQueue). To reduce the amount of required memory, this is performed on a per-file basis. The BPPARAM argument allows to specify how and if parallel processing should be used. Information from individual files will be processed in parallel (one process per original file).

The method returns a `list` of numeric M/Z values. Each list element represents the values from one spectrum.

peaksCount `peaksCount(signature(object="OnDiskMSnExp",scans="numeric"), BPPARAM=bpparam())`: extrac the peaks count from each spectrum in the object. Depending on the eventually present `ProcessingStep` objects in the `spectraProcessingQueue` raw data will be loaded to calculate the peaks count. If no steps are present, the data is extracted from the `featureData`. Optional argument `scans` allows to specify the index of specific spectra from which the count should be returned. The `BPPARAM` argument can be used to define how and if parallel processing should be used. Information from individual files will be processed in parallel (one process per original file).

Returns a numeric vector with names corresponding to the spectrum names.

polarity `polarity(signature(object="OnDiskMSnExp"))`: returns a numeric vector with the polarity of the individual spectra in the data set. The relevant information is extracted from the `featureData`.

rtime `rtime(signature(object="OnDiskMSnExp"))`: extrac the retention time of the individual spectra in the data set (from the `featureData`).

Returns a numeric vector with names corresponding to the spectrum names.

scanIndex `scanIndex(signature(object="OnDiskMSnExp"))`: get the spectra scan indices within the respective file. The relevant information is extracted from the object's `featureData` slot. Returns a numeric vector of indices with names corresponding to the spectrum names.

smoothed,smoothed<- `smoothed(signature(object="OnDiskMSnExp", msLevel. = "numeric"))`: whether individual spectra are smoothed or unsmoothed. The relevant information is extracted from the object's `featureData` slot. Returns a logical vector with names corresponding to the spectrum names. Use `smoothed(object) <- value` to update the information, with value being a logical vector of length equal to the number of spectra in the experiment.

spectra `spectra(signature(object="OnDiskMSnExp"), BPPARAM=bpparam())`: extract spectrum data from the individual files. This causes the spectrum data to be read from the original raw files. After that all processing steps defined in the `spectraProcessingQueue` are applied to it. The results are then returned as a `list` of `Spectrum1` objects.

The `BPPARAM` argument can be used to define how and if parallel processing should be used. Information from individual files will be processed in parallel (one process per file). Note: extraction of selected spectra results in a considerable processing speed and should thus be preferred over whole data extraction.

Returns a `list` of `Spectrum1` objects with names corresponding to the spectrum names.

tic `tic(signature(object="OnDiskMSnExp"), initial = TRUE, BPPARAM = bpparam())`: get the total ion current (TIC) of each spectrum in the data set. If `initial = TRUE`, the information is extracted from the object's `featureData` and represents the tic provided in the header of the original raw data files. For `initial = FALSE`, the TIC is calculated from the actual intensity values in each spectrum after applying all data manipulation methods (if any).

See also https://github.com/lgatto/MSnbase/issues/332 for more details.

`BPPARAM` parameter: see `spectra` method above.

Returns a numeric vector with names corresponding to the spectrum names.

bpi `bpi(signature(object="OnDiskMSnExp"), initial = TRUE, BPPARAM = bpparam())`: get the base peak intensity (BPI), i.e. the maximum intensity from each spectrum in the data set. If `initial = TRUE`, the information is extracted from the object's `featureData` and represents the bpi provided in the header of the original raw data files. For `initial = FALSE`, the

BPI is calculated from the actual intensity values in each spectrum after applying all eventual data manipulation methods.

See also https://github.com/lgatto/MSnbase/issues/332 for more details.

BPPARAM parameter: see spectra method above.

Returns a numeric vector with names corresponding to the spectrum names.

featureNames tic(signature(object="OnDiskMSnExp")): return a character of length length(object) containing the feature names. A replacement method is also available.

spectrapply spectrapply(signature(object = "OnDiskMSnExp"), FUN = NULL,BPPARAM = bpparam(), ...): applies the function FUN to each spectrum passing additional parameters in ... to that function and return its results. For FUN = NULL it returns the list of spectra (same as a call to spectra). Parameter BPPARAM allows to specify how and if parallel processing should be enabled.

Returns a list with the result for each of spectrum.

Data manipulation methods

(in alphabetical order) See also methods for MSnExp or pSet objects. In contrast to the same-named methods for pSet or MSnExp classes, the actual data manipulation is not performed immediately, but only on-demand, e.g. when intensity or M/Z values are loaded.

clean clean(signature(object="OnDiskMSnExp"), all=TRUE, verbose=TRUE): add an *clean* processing step to the lazy processing queue of the OnDiskMSnExp object. The clean command will only be executed when spectra information (including M/Z and intensity values) is requested from the OnDiskMSnExp object. Optional arguments to the methods are all=TRUE and verbose=TRUE.

The method returns an OnDiskMSnExp object.

For more details see documentation of the clean method.

normalize normalize(signature(object="OnDiskMSnExp"), method=c("max","sum"), ...): add a normalize processing step to the lazy processing queue of the returned OnDiskMSnExp object.

The method returns an OnDiskMSnExp object.

For more details see documentation of the normalize method.

removePeaks removePeaks(signature(object="OnDiskMSnExp"), t="min", verbose=TRUE): add a removePeaks processing step to the lazy processing queue of the returned OnDiskMSnExp object.

The method returns an OnDiskMSnExp object.

For more details see documentation of the removePeaks method.

trimMz trimMz(signature(object="OnDiskMSnExp", mzlim="numeric"),...): add a trimMz processing step to the lazy processing queue of the returned OnDiskMSnExp object.

The method returns an OnDiskMSnExp object.

For more details see documentation of the trimMz method.

Other methods and functions

validateOnDiskMSnExp validateOnDiskMSnExp(signature(object = "OnDiskMSnExp")): validates an OnDiskMSnExp object and all of its spectra. In addition to the *standard* validObject

method, this method reads also all spectra from the original files, applies eventual processing steps and evaluates their validity.

`as(from, "MSnExp")` Converts the `OnDiskMSnExp` object `from`, to an in-memory `MSnExp`. Also available as an S3 method `as.MSnExp()`.

Author(s)

Johannes Rainer <johannes.rainer@eurac.edu>

See Also

`pSet`, `MSnExp`, `readMSData`

Examples

```
## Get some example mzML files
library(msdata)
mzfiles <- c(system.file("microtofq/MM14.mzML", package="msdata"),
      system.file("microtofq/MM8.mzML", package="msdata"))
## Read the data as an OnDiskMSnExp
odmse <- readMSData(mzfiles, msLevel=1, centroided = TRUE)

## Get the length of data, i.e. the total number of spectra.
length(odmse)

## Get the MS level
head(msLevel(odmse))

## Get the featureData, use fData to return as a data.frame
head(fData(odmse))

## Get to know from which file the spectra are
head(fromFile(odmse))

## And the file names:
fileNames(odmse)

## Scan index and acquisitionNum
head(scanIndex(odmse))
head(acquisitionNum(odmse))

## Extract the spectra; the data is retrieved from the raw files.
head(spectra(odmse))

## Extracting individual spectra or a subset is much faster.
spectra(odmse[1:50])

## Alternatively, we could also subset the whole object by spectra and/or samples:
subs <- odmse[rtime(odmse) >= 2 & rtime(odmse) <= 20, ]
fileNames(subs)
rtime(subs)
```

```
## Extract intensities and M/Z values per spectrum; the methods return a list,
## each element representing the values for one spectrum.
ints <- intensity(odmse)
mzs <- mz(odmse)

## Return a data.frame with mz and intensity pairs for each spectrum from the
## object
res <- spectrapply(odmse, FUN = as, Class = "data.frame")

## Calling removePeaks, i.e. setting intensity values below a certain threshold to 0.
## Unlike the name suggests, this is not actually removing peaks. Such peaks with a 0
## intensity are then removed by the "clean" step.
## Also, the manipulations are not applied directly, but put into the "lazy"
## processing queue.
odmse <- removePeaks(odmse, t=10000)
odmse <- clean(odmse)

## The processing steps are only applied when actual raw data is extracted.
spectra(odmse[1:2])

## Get the polarity of the spectra.
head(polarity(odmse))

## Get the retention time of all spectra
head(rtime(odmse))

## Get the intensities after removePeaks and clean
intsAfter <- intensity(odmse)

head(lengths(ints))
head(lengths(intsAfter))

## The same for the M/Z values
mzsAfter <- intensity(odmse)
head(lengths(mzs))
head(lengths(mzsAfter))


## Centroided or profile mode
f <- msdata::proteomics(full.names = TRUE,
   pattern = "MS3TMT11.mzML")
odmse <- readMSData(f, mode = "onDisk")
validObject(odmse)
odmse[[1]]

table(isCentroidedFromFile(odmse), msLevel(odmse))

## centroided status could be set manually
centroided(odmse, msLevel = 1) <- FALSE
centroided(odmse, msLevel = 2) <- TRUE
centroided(odmse, msLevel = 3) <- TRUE

## or when reading the data
```

```
odmse2 <- readMSData(f, centroided = c(FALSE, TRUE, TRUE),
       mode = "onDisk")
table(centroided(odmse), msLevel(odmse))

## Filtering precursor scans

head(acquisitionNum(odmse))
head(msLevel(odmse))

## Extract all spectra stemming from the first MS1 spectrum
(from1 <- filterPrecursorScan(odmse, 21945))
table(msLevel(from1))


## Extract the second sepctrum's parent (MS1) and children (MS3)
## spectra
(from2 <- filterPrecursorScan(odmse, 21946))
table(msLevel(from2))
```

`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 available, the Median Absolute Deviation (`method = "MAD"`) and Friedman's Super Smoother (`method = "SuperSmoother"`), as implemented in the `MALDIquant::detectPeaks` and `MALDIquant::estimateNoise` functions respectively.

The method supports also to optionally *refine* the m/z value of the identified centroids by considering data points that belong (most likely) to the same mass peak. The m/z value is calculated as an intensity weighted average of the m/z values within the peak region. How the peak region is defined depends on the method chosen:

`refineMz = "kNeighbors"`: m/z values (and their respective intensities) of the`2 * k` closest signals to the centroid are used in the intensity weighted average calculation. The number of neighboring signals can be defined with the argument `k`.

`refineMz = "descendPeak"`: the peak region is defined by descending from the identified centroid/peak on both sides until the measured signal increases again. Within this defined region all measurements with an intensity of at least `signalPercentage` of the centroid's intensity are used to calculate the refined m/z. By default the descend is stopped when the first signal that is equal or larger than the last observed one is encountered. Setting `stopAtTwo = TRUE`, two consecutively increasing signals are required.

By default (`refineMz = "none"`, simply the m/z of the largest signal (the identified centroid) is reported. See below for examples.

Methods

`signature(x = "MSnExp", halfWindowSize = "integer", method = "character", SNR = "numeric", verbose = "logi`
: 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. `refineMz` allows to choose a method for an optional centroid m/z refinement (see description for more details). Choises are `"none"` (default, no m/z refinement), `"kNeighbors"` and `"descendPeak"`. The arguments `...` are passed to the noise estimator or m/z refinement functions. For the noise estimator functions, currenlty only the `method = "SuperSmoother"` accepts additional arguments, e.g. `span`. Please see `supsmu` for details. `refineMethod = "kNeighbors"` supports additional argument `k` and `refineMethod = "descendPeak"` arguments `signalPercentage` and `stopAtTwo`. See description above for more 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> with contributions from Johannes Rainer.

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`, `estimateNoise` and `trimMz` for other spectra processing methods.

Examples

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

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


## Examples for refineMz:
```

```
ints <- c(5, 3, 2, 3, 1, 2, 4, 6, 8, 11, 4, 7, 5, 2, 1, 0, 1, 0, 1, 1, 1, 0)
mzs <- 1:length(ints)
sp1 <- new("Spectrum1", intensity = ints, mz = mzs, centroided = FALSE)
plot(mz(sp1), intensity(sp1), type = "h")

## Without m/z refinement:
sp2 <- pickPeaks(sp1)
points(mz(sp2), intensity(sp2), col = "darkgrey")
## Using k = 1, closest signals
sp3 <- pickPeaks(sp1, refineMz = "kNeighbors", k = 1)
points(mz(sp3), intensity(sp3), col = "green", type = "h")

## Using descendPeak requiring at least 50% or the centroid's intensity
sp4 <- pickPeaks(sp1, refineMz = "descendPeak", signalPercentage = 50)
points(mz(sp4), intensity(sp4), col = "red", type = "h")
```

`plot-methods` *Plotting 'MSnExp' and 'Spectrum' object(s)*

Description

These methods provide the functionality to plot mass spectrometry data provided as `MSnExp`, `OnDiskMSnExp` or `Spectrum` objects. Most functions plot mass spectra M/Z values against 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.

Most methods make use the `ggplot2` system in which case 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 fragement 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

`plot(signature(x = "MSnExp", y = "missing"), type = c("spectra", "XIC"), reporters = "ReporterIons", full`

For `type = "spectra"`: 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'.

For `type = "XIC"`: Plots a combined plot of retention time against m/z values and retention time against largest signal per spectrum for each file. Data points are colored by intensity. The lower part of the plot represents the location of the individual signals in the retention time - m/z space, the upper part the base peak chromatogram of the data (i.e. the largest signal for each spectrum). This plot type is restricted to MS level 1 data and is most useful for LC-MS data. Ideally, the `MSnExp` (or `OnDiskMSnExp`) object should be filtered first using the `filterRt` and `filterMz` functions to narrow on an ion of interest. See examples below. This plot uses base R plotting. Additional arguments to the `plot` function can be passed with `...`.

Additional arguments for `type = "XIC"` are:

`col` color for the border of the points. Defaults to `col = "grey"`.

`colramp` color function/ramp to be used for the intensity-dependent background color of data points. Defaults to `colramp = topo.colors`.

`grid.color` color for the grid lines. Defaults to `grid.color = "lightgrey"`; use `grid.color = NA` to disable grid lines altogether.

`pch` point character. Defaults to `pch = 21`.

`...` additional parameters for the low-level `plot` function.

`plot(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.

`plot(signature(x = "Spectrum2", y = "character"), orientation = "numeric", add = "logical", col = "charact`

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 values are `orientation=1`, `add=FALSE`, `col="#74ADD1"`, `pch=NA`, `xlab="m/z"`, `ylab="intensity"`, `ylim=c(0, 1)`, `tolerance=25e-6`, relative=TRUE, 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, Johannes Rainer 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.

Chromatogram for plotting of chromatographic data.

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)

## Load profile-mode LC-MS files
library(msdata)
od <- readMSData(dir(system.file("sciex", package = "msdata"),
                     full.names = TRUE), mode = "onDisk")
## Restrict the MS data to signal for serine
serine <- filterMz(filterRt(od, rt = c(175, 190)), mz = c(106.04, 106.06))
plot(serine, type = "XIC")

## Same plot but using heat.colors, rectangles and no point border
plot(serine, type = "XIC", pch = 22, colramp = heat.colors, col = NA)
```

---

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=57.02146)`). Additional you can specifiy the type of `neutralLoss` (default: `PSMatch::defaultNeutralLoss()`). See `calculateFragments` for details.

There are a lot of graphical arguments available to control the representation of the peaks and fragments. Use `peaks.pch` to set the character on top of the peaks (default: `peaks.pch=19`). In a similar way you can set the line width `peaks.lwd=1` and the magnification `peaks.cex=0.5` of the peaks. The size of the fragment/legend labels could be set using `fragments.cex=0.75` or `legend.cex` respectively. See `par` for details about graphical parameters in general.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

See Also

More spectrum plotting available in `plot.Spectrum`.

More details about fragment calculation: `calculateFragments`.

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, centroided.=FALSE)

## 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

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

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 and Guangchuang Yu

References

Foster JM, Degroeve S, Gatto L, Visser M, Wang R, Griss J, Apweiler R, Martens L. "A posteriori 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)` specifiying 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

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

Examples

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

ProcessingStep-class *Simple processing step class*

Description

The ProcessingStep class is a simple object to encapsule all relevant information of a data analysis processing step, i.e. the function name and all arguments.

Details

Objects of this class are mainly used to record all possible processing steps of an OnDiskMSnExp object for later *lazy execution*.

Objects from the Class

Objects can be created by calls of the form `new("ProcessingStep",...)` or using the `ProcessingStep` constructor function.

Slots

`FUN`: The function name to be executed as a character string.

`ARGS`: A named `list` with all arguments to the function.

Methods and functions

executeProcessingStep(object, ...) Execute the processing step `object`. Internally this calls `do.call` passing all arguments defined in the `ProcessingStep` `object` along with potential additional arguments in `...` to the function `object@FUN`.

Extends

Class "Versioned", directly.

Author(s)

Johannes Rainer <johannes.rainer@eurac.edu>

See Also

OnDiskMSnExp

Examples

```
## Define a simple ProcessingStep
procS <- ProcessingStep("sum", list(c(1, 3, NA, 5), na.rm= TRUE))

executeProcessingStep(procS)
```

---

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"`).

`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.

`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.

`experimentData`: Object of class `"MIAPE"`, containing details of experimental methods. See `experimentData` for more details.

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

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

`.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.

$ `signature(x = "pSet")`: directly access a specific sample annotation column from the `pData`.

$<- `signature(x = "pSet")`: replace or add a sample annotation column in the `pData`.

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 `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.

smoothed `signature(object = "pSet")`: Indicates whether individual spectra are smoothed ('TRUE') of unsmoothed ('FALSE'). Use `smoothed(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.

pData<- `signature(x = "pSet", value)`: Replace sample data information with `value`, value being a `data.frame`.

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

phenoData<- `signature(x = "pSet", value)`: Replace sample data information with `value`. `value` can be a `data.frame` or an `AnnotatedDataFrame`.

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`. A replacement method is also available.

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).

spectrapply `spectrapply(signature(object = "pSet"), FUN = NULL, BPPARAM = bpparam(), ...)`: applies the function `FUN` to each spectrum passing additional parameters in `...` to that function and return its results. For `FUN = NULL` it returns the list of spectra (same as a call to `spectra`). Parameter `BPPARAM` allows to specify how and if parallel processing should be enabled.

Returns a list with the result for each of spectrum.

isolationWindowLowerMz `isolationWindowLowerMz(object = "pSet")`: return the lower m/z boundary for the isolation window. Note that this method is at present only available for `OnDiskMSnExp` objects.

isolationWindowUpperMz `isolationWindowUpperMz(object = "pSet")`: return the upper m/z boundary for the isolation window. Note that this method is at present only available for `OnDiskMSnExp` objects.

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

Author(s)

Laurent Gatto

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 coefficients 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 (for example lot RH239932)

.	-2	-1	1	2
126	0.0	0.0	5.0 (127C)	0.0 (128C)
127N	0.0	0.2	5.8 (128N)	0.0 (129N)
127C	0.0	0.3 (126)	4.8 (128C)	0.0 (129C)
128N	0.0	0.4 (127N)	4.1 (129N)	0.0 (130N)
128C	0.0 (126)	0.6 (127C)	3.0 (129C)	0.0 (130C)
129N	0.0 (127N)	0.8 (128N)	3.5 (130N)	0.0 (131)
129C	0.0 (127C)	1.4 (128C)	2.4 (130C)	0.0
130N	0.1 (128N)	1.5 (129N)	2.4 (131)	3.2
130C	0.0 (128C)	1.7 (129C)	1.8	0.0
131	0.2 (129N)	2.0 (130N)	2.2	0.0

(Note that a previous example, taken from lot PB199188A, contained a typo.)
the impurity correction matrix is

.	126	127N	127C	128N	128C	129N	129C	130N	130C	131
% reporter 126	0.950	0.000	0.050	0.000	0.000	0.000	0.000	0.000	0.000	0.000
% reporter 127N	0.000	0.940	0.000	0.058	0.000	0.000	0.000	0.000	0.000	0.000
% reporter 127C	0.003	0.000	0.949	0.000	0.048	0.000	0.000	0.000	0.000	0.000
% reporter 128N	0.000	0.004	0.000	0.955	0.000	0.041	0.000	0.000	0.000	0.000
% reporter 128C	0.000	0.000	0.006	0.000	0.964	0.000	0.030	0.000	0.000	0.000
% reporter 129N	0.000	0.000	0.000	0.008	0.000	0.957	0.000	0.035	0.000	0.000
% reporter 129C	0.000	0.000	0.000	0.000	0.014	0.000	0.962	0.000	0.024	0.000
% reporter 130N	0.000	0.000	0.000	0.001	0.000	0.015	0.000	0.928	0.000	0.024
% reporter 130C	0.000	0.000	0.000	0.000	0.000	0.000	0.017	0.000	0.965	0.000
% reporter 131	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.020	0.000	0.956

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.
pepseq	A character giving the peptide sequence column in the feature data. Default is "sequence".
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 "DatabaseAccess") and the protein lengths (plength, with default value "DBseqLength"). 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 and Sebastian Gibb

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)$DatabaseAccess

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

```
processingData(si)
exprs(si)

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

---

readMgfData *Import mgf files as 'MSnExp' instances.*

Description

Reads a mgf file and generates an "MSnExp" object.

Usage

```
readMgfData(filename, pdata = NULL, centroided = TRUE, smoothed = FALSE,
verbose = isMSnbaseVerbose(), cache = 1)
```

Arguments

filename	character vector with file name to be read.
pdata	an object of class "AnnotatedDataFrame".
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 and Laurent Gatto

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
f <- dir(system.file(package="MSnbase",dir="extdata"),
         full.name=TRUE,
         pattern="test.mgf")
(x <- readMgfData(f))
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. = NULL,
```

```
  verbose = isMSnbaseVerbose(),
  centroided. = NA,
  smoothed. = NA,
  cache. = 1L,
  mode = c("inMemory", "onDisk")
)
```

Arguments

files	A character with file names to be read and parsed.
pdata	An object of class AnnotatedDataFrame or NULL (default).
msLevel.	MS level spectra to be read. In inMemory mode, use 1 for MS1 spectra or any larger numeric for MSn spectra. Default is 2 for InMemory mode. onDisk mode supports multiple levels and will, by default, read all the data.
verbose	Verbosity flag. Default is to use isMSnbaseVerbose().
centroided.	A logical, indicating whether spectra are centroided or not. Default is NA in which case the information is extracted from the raw file (for mzML or mzXML files). In onDisk, it can also be set for different MS levels by a vector of logicals, where the first element is for MS1, the second element is for MS2, ... See OnDiskMSnExp for an example.
smoothed.	A logical indicating whether spectra already smoothed or not. Default is NA.
cache.	Numeric indicating caching level. Default is 0 for MS1 and 1 MS2 (or higher). Only relevant for inMemory mode.
mode	On of "inMemory" (default) or "onDisk". The former loads the raw data in memory, while the latter only generates the object and the raw data is accessed on disk when needed. See the *benchmarking* vignette for memory and speed implications.

Details

When using the inMemory mode, the whole MS data is read from file and kept in memory as Spectrum objects within the MSnExp'es assayData slot.

To reduce the memory footpring especially for large MS1 data sets it is also possible to read only selected information from the MS files and fetch the actual spectrum data (i.e. the M/Z and intensity values) only on demand from the original data files. This can be achieved by setting mode = "onDisk". The function returns then an OnDiskMSnExp object instead of a MSnExp object.

Value

An MSnExp object for inMemory mode and a OnDiskMSnExp object for onDisk mode.

Note

readMSData uses normalizePath to replace relative with absolute file paths.

Author(s)

Laurent Gatto

See Also

readMgfData() to read mgf peak lists.

Examples

```
file <- dir(system.file(package = "MSnbase", dir = "extdata"),
            full.name = TRUE,
            pattern = "mzXML$")
mem <- readMSData(file, mode = "inMemory")
mem
dsk <- readMSData(file, mode = "onDisk")
dsk
```

---

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.

Note that when using readMSnSet2, one should not set rownames as additional argument to defined feature names. It is ignored and used to set fnames if not provided otherwise.

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, stringsAs
       featureDataArgs = list(sep = sep, header = header, row.names = row.names, quote = quote, stringsA
       experimentDataArgs = list(sep = sep, header = header, row.names = row.names, quote = quote, strin
           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.

`exprsFile` (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 or a `data.frame` (new in version 1.19.8). Default, when `file` is a `character`, 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.

Passing a `data.frame` can be particularly useful if the spreadsheet is in Excel format. The appropriate sheet can first be read into R as a `data.frame` using, for example `readxl::read_excel`, and then pass it to `readMSnSet2`.

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 if 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 dislpay the column names are see by R.
fnames	An optional character or numeric of length 1 indicating the column to be used as feature names.

Value

An instance of the MSnSet class.

Author(s)

Laurent Gatto <laurent.gatto@uclouvain.be>

See Also

The grepEcols and getEcols helper functions to identify the ecol values. The MSnbase-io vignette illustrates these functions in detail. It can be 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)

library("pRolocdata")
f0 <- dir(system.file("extdata", package = "pRolocdata"),
          full.names = TRUE,
          pattern = "Dunkley2006")
basename(f0)
res <- readMSnSet2(f0, ecol = 5:20)
res
head(exprs(res)) ## columns 5 to 20
head(fData(res)) ## other columns
```

---

readMzIdData *Import peptide-spectrum matches*

Description

Reads as set of mzId files containing PSMs an generates a data.frame.

Usage

```
readMzIdData(files)
```

Arguments

files	A character of mzid files.

Details

This function uses the functionality provided by the mzR package to access data in the mzId files. An object of class mzRident can also be coerced to a data.frame using as(, "data.frame").

Value

A data.frame containing the PSMs stored in the mzId files.

Author(s)

Laurent Gatto

See Also

filterIdentificationDataFrame() to filter out unreliable PSMs.

Examples

```
f <- MsDataHub::TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.20141210.mzid()
readMzIdData(f)
```

---

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 = isMSnbaseVerbose()
)
```

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://raw.githubusercontent.com/HUPO-PSI/mzTab/master/examples/1_0-Proteomics-Release/PRIDE_Exp_

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 = isMSnbaseVerbose())
```

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.

---

readSRMData *Read SRM/MRM chromatographic data*

Description

The readSRMData function reads MRM/SRM data from provided *mzML* files and returns the results as a MChromatograms() object.

Usage

```
readSRMData(files, pdata = NULL)
```

Arguments

files	character with the files containing the SRM/MRM data.
pdata	data.frame or AnnotatedDataFrame with file/sample descriptions.

Details

readSRMData supports reading chromatogram entries from *mzML* files. If multiple files are provided the same precursor and product m/z for SRM/MRM chromatograms are expected across files. The number of columns of the resulting MChromatograms() object corresponds to the number of files. Each row in the MChromatograms object is supposed to contain chromatograms with same polarity, precursor and product m/z. If chromatograms with redundant polarity, precursor and product m/z values and precursor collision energies are found, they are placed into multiple consecutive rows in the MChromatograms object.

Value

A MChromatograms() object. See details above for more information.

Note

readSRMData reads only SRM/MRM chromatogram data, i.e. chromatogram data from mzML files with precursorIsolationWindowTargetMZ and productIsolationWindowTargetMZ attributes. Total ion chromatogram data is hence not extracted.

The number of features and hence rows of the resulting MChromatograms object depends on the total list of unique precursor and product m/z isolation windows (and precursor collision energies) found across all input files. In cases in which not each file has chromatgraphic data for the same polarity, precursor m/z, product m/z and collision energy, an empty Chromatogram() object is reported for the specific precursor and product m/z combination of the respective file (and a warning is thrown).

Author(s)

Johannes Rainer

Examples

```
## Read an example MRM/SRM data
library(msdata)
fl <- proteomics(full.names = TRUE, pattern = "MRM")

## Read the data
mrm <- readSRMData(fl)

## The data is represented as a MChromatograms object, each column
## containing the data from one input file
mrm

## Access the polarity for each chromatogram (row)
polarity(mrm)

## Access the precursor m/z. The result is returned as a matrix with
## columns representing the minimum and maximum m/z (will be identical in
## most cases).
precursorMz(mrm)

## Access the product m/z.
productMz(mrm)

## Plot one chromatogram
plot(mrm[1, ])
```

reduce,data.frame-method
Reduce a data.frame

Description

Reduce a data.frame so that the (primary) key column contains only unique entries and other columns pertaining to that entry are combined into semicolon-separated values into a single row/observation.

An important side-effect of reducing a `data.frame` is that all columns other than the key are converted to characters when they are collapsed to a semi-column separated value (even if only one value is present) as soon as one observation of transformed.

Usage

```
## S4 method for signature 'data.frame'
reduce(x, key, sep = ";")
```

Arguments

`x`	A `data.frame`.
`key`	The column name (currenly only one is supported) to be used as primary key.
`sep`	The separator. Default is `;`.

Value

A reduced `data.frame`.

Author(s)

Laurent Gatto

Examples

```
dfr <- data.frame(A = c(1, 1, 2),
                  B = c("x", "x", "z"),
                  C = LETTERS[1:3])
dfr
dfr2 <- reduce(dfr, key = "A")
dfr2
## column A used as key is still num
str(dfr2)
dfr3 <- reduce(dfr, key = "B")
dfr3
## A is converted to chr; B remains factor
str(dfr3)
dfr4 <- data.frame(A = 1:3,
                   B = LETTERS[1:3],
                   C = c(TRUE, FALSE, NA))
```

```
## No effect of reducing, column classes are maintained
str(reduce(dfr4, key = "B"))
```

`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` instances.

Author(s)

Laurent Gatto

See Also

`MSnExp` and `MSnSet`.

Examples

```
quantFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "mzXML$")
identFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "dummyiTRAQ.mzid")
msexp <- readMSData(quantFile)
msexp <- addIdentificationData(msexp, identFile)
fData(msexp)$sequence
length(msexp)

## using default fcol
msexp2 <- removeNoId(msexp)
length(msexp2)
fData(msexp2)$sequence

## using keep
print(fvarLabels(msexp))
(k <- fData(msexp)$'MS.GF.EValue' > 75)
k[is.na(k)] <- FALSE
k
msexp3 <- removeNoId(msexp, keep = k)
length(msexp3)
fData(msexp3)$sequence
```

`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, msLevel. = "numeric")` 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. `msLevel.` defines the level of the spectrum, and if `msLevel(object) != msLevel.`, cleaning is ignored. Only relevant when called from `OnDiskMSnExp` and is only relevant for developers.

Returns a new `Spectrum` instance.

Author(s)

Laurent Gatto

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),
           centroided = FALSE)
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))
stopifnot(identical(intensity(sp), 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

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 explicitely, 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

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, Schäfer 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]
```

---

selectFeatureData *Select feature variables of interest*

Description

Select feature variables to be retained.

`requiredFvarLabels` returns a `character` vector with the required feature data variable names (`fvarLabels`, i.e. the column names in the `fData` `data.frame`) for the specified object.

Usage

```
selectFeatureData(object, graphics = TRUE, fcol)

requiredFvarLabels(x = c("OnDiskMSnExp", "MSnExp", "MSnSet"))
```

Arguments

object	An MSnSet, MSnExp or OnDiskMSnExp.
graphics	A logical (default is TRUE) indicating whether a shiny application should be used if available. Otherwise, a text menu is used. Ignored if k is not missing.
fcol	A numeric, logical or character of valid feature variables to be passed directly.
x	character(1) specifying the class name for which the required feature data variable names should be returned.

Value

For selectFeatureData: updated object containing only selected feature variables.

For requiredFvarLabels: character with the required feature variable names.

Author(s)

Laurent Gatto

Examples

```
library("pRolocdata")
data(hyperLOPIT2015)
## 5 first feature variables
x <- selectFeatureData(hyperLOPIT2015, fcol = 1:5)
fvarLabels(x)
## Not run:
## select via GUI
x <- selectFeatureData(hyperLOPIT2015)
fvarLabels(x)

## End(Not run)

## Subset the feature data of an OnDiskMSnExp object to the minimal
## required columns
f <- system.file("microtofq/MM14.mzML", package = "msdata")
od <- readMSData(f, mode = "onDisk")

## what columns do we have?
fvarLabels(od)

## Reduce the feature data data.frame to the required columns only
od <- selectFeatureData(od, fcol = requiredFvarLabels(class(od)))
fvarLabels(od)
```

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"`. `"halfWindowSize"` controls the window size of the filter. The resulting window size is `2 * halfWindowSize + 1`. The best size differs depending on the selected `method`. For `method = "SavitzkyGolay"` it should be lower than *FWHM* of the peaks (full width at half maximum; please find details in Bromba and Ziegler 1981). The arguments `...` are passed to the internal functions. For `method="MovingAverage"` there is an additional `weighted` argument (default: `FALSE`) to indicate if the average should be equal weight (default) or if it should have weights depending on the distance from the center as calculated as `1/2^abs(-halfWindowSize:halfWindowSize)` with the sum of all weigths normalized to 1. For `method="SavitzkyGolay"` an additonal argument is `polynomialOrder` (default: 3). It controls 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 spectrometry 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.

In version 1.19.12, the polarity slot has been added to this class (previously in "Spectrum1").

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.

polarity: Object of class "integer" indicating the polarity if the ion.

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, as reported in the original raw data file.

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'). Default is NA.

smoothed: Object of class "logical" indicating if instance is smoothed ('TRUE') of unsmoothed ('FALSE'). Default is NA.

`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`), in profile mode (`FALSE`), or unkown (`NA`).

`isCentroided(object, k=0.025, qtl=0.9)` A heuristic assessing if a spectrum is in profile or centroided mode. The function takes the `qtl`th quantile top peaks, then calculates the difference between adjacent M/Z value and returns `TRUE` if the first quartile is greater than `k`. (See `MSnbase:::.isCentroided` for the code.) The function has been tuned to work for MS1 and MS2 spectra and data centroided using different peak picking algorithms, but false positives can occur. See https://github.com/lgatto/MSnbase/issues/131 for details. It should however be safe to use is at the experiment level, assuming that all MS level have the same mode. See `class?MSnExp` for an example.

`smoothed(object)` Indicates whether spectrum is smoothed (`TRUE`) or not (`FALSE`).

`centroided(object) <- value` Sets the `centroided` status of the spectrum object.

`smoothed(object) <- value` Sets the `smoothed` 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. This is the total ion current as originally reported in the raw data file. To get the current total ion count, use `ionCount`.

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(x = "Spectrum", y = "Spectrum")`: Compares spectra. See `compareSpectra` documentation for more details and examples.

estimateNoise `signature(object = "Spectrum")`: Estimates the noise in a profile spectrum. See `estimateNoise` 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

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

See the "Spectrum" class for inherited slots.

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

See Also

Virtual super-class "Spectrum", "Spectrum2" for MS2 spectra and "MSnExp" for a full experiment container.

Spectrum2-class *The "Spectrum2" Class for MSn Spectra*

Description

Spectrum2 extends the "Spectrum" class and introduces several MS2 specific attributes in addition to the slots in "Spectrum". Since version 1.99.2, this class is used for any MS levels > 1. Spectrum2 are not created directly but are contained in the assayData slot of an "MSnExp".

In version 1.19.12, the polarity slot had been added to the "Spectrum" class (previously in "Spectrum1"). Hence, "Spectrum2" objects created prior to this change will not be valid anymore, since they will miss the polarity slots. Object can be appropriately updated using the updateObject method.

Slots

See the "Spectrum" class for inherited 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.

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

See Also

Virtual super-class "Spectrum", "Spectrum1" for MS1 spectra and "MSnExp" for a full experiment container.

TMT6 *TMT 6/10-plex sets*

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. A second TMT6b has slightly different values.

The TMT10 instance corresponds to the 10-plex version. There are spectific HCD (TMT10HCD, same as TMT10) and ETD (TMT10ETD) sets.

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
TMT6b
TMT7
TMT7b
TMT10
TMT10ETD
TMT10HCD
TMT11
TMT11HCD
```

References

Thompson A, Schäfer 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 `mz` argument, such that MZ values <= `min(mz)` and MZ values >= `max(mz)` are trimmed away.

Methods

`signature(object = "MSnExp", mz = "numeric", msLevel. = "numeric")` Trims all spectra in `MSnExp` object according to `mz`. If `msLevel.` is defined, then only spectra of that level are trimmer.

`signature(object = "Spectrum", mz = "numeric", msLevel. = "numeric")` Trims the `Spectrum` object and retruns a new trimmed object. `msLevel.` defines the level of the spectrum, and if `msLevel(object) != msLevel.`, cleaning is ignored. Only relevant when called from `OnDiskMSnExp` and is only relevant for developers.

Author(s)

Laurent Gatto

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 <- filterMz(itraqdata, c(113, 117))
range(mz(itraqdata))
range(mz(itraqdata2))
processingData(itraqdata2)
```

updateObject-methods *Update MSnbase objects*

Description

Methods for function updateObject for objects from the MSnbase package. See updateObject for details.

Methods

signature(object = "MSnExp") Update the MSnExp object to the latest class version

signature(object = "Spectrum") Update the Spectrum object (and it's sub-classes Spectrum1 and Spectrum2) to the latest class version.

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 overwritten.
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

```
data(itraqdata)

f <- tempfile()

writeMgfData(itraqdata, con = f)

itraqdata2 <- readMgfData(f)

## 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)

all.equal(as.data.frame(itraqdata2[[1]]),
          as.data.frame(itraqdata[[1]]))

all.equal(as.data.frame(itraqdata2[[3]]),
          as.data.frame(itraqdata[[11]]))

all(featureNames(itraqdata2) == featureNames(itraqdata))
```

---

writeMSData,MSnExp,character-method

Write MS data to mzML or mzXML files

Description

The writeMSData,MSnExp and writeMSData,OnDiskMSnExp saves the content of a MSnExp or OnDiskMSnExp object to MS file(s) in either *mzML* or *mzXML* format.

Usage

```
## S4 method for signature 'MSnExp,character'
writeMSData(
  object,
  file,
  outformat = c("mzml", "mzxml"),
```

```
  merge = FALSE,
  verbose = isMSnbaseVerbose(),
  copy = FALSE,
  software_processing = NULL
)
```

Arguments

object	OnDiskMSnExp or MSnExp object.
file	character with the file name(s). Its length has to match the number of samples/files of x.
outformat	character(1) defining the format of the output files. Default output format is "mzml".
merge	logical(1) whether the data should be saved into a single *mzML* file. Default is merge = FALSE, i.e. each sample is saved to a separate file. **Note**: merge = TRUE is not yet implemented.
verbose	logical(1) if progress messages should be displayed.
copy	logical(1) if metadata (data processings, original file names etc) should be copied from the original files. See details for more information.
software_processing	optionally provide specific data processing steps. See documentation of the software_processing parameter of mzR::writeMSData().

Details

The writeMSData method uses the *proteowizard* libraries through the mzR package to save the MS data. The data can be written to *mzML* or *mzXML* files with or without copying additional metadata information from the original files from which the data was read by the readMSData() function. This can be set using the copy parameter. Note that copy = TRUE requires the original files to be available and is not supported for input files in other than mzML or mzXML format. All metadata related to the run is copied, such as instrument information, data processings etc. If copy = FALSE only processing information performed in R (using MSnbase) are saved to the mzML file.

Currently only spectrum data is supported, i.e. if the original mzML file contains also chromatogram data it is not copied/saved to the new mzML file.

Note

General spectrum data such as total ion current, peak count, base peak m/z or base peak intensity are calculated from the actual spectrum data before writing the data to the files.

For MSn data, if the OnDiskMSnExp or MSnExp does not contain also the precursor scan of a MS level > 1 spectrum (e.g. due to filtering on the MS level) precursorScanNum is set to 0 in the output file to avoid potentially linking to a wrong spectrum.

The exported mzML file *should* be valid according to the mzML 1.1.2 standard. For exported mzXML files it can not be guaranteed that they are valid and can be opened with other software than mzR/MSnbase.

Author(s)

Johannes Rainer

`writeMzTabData` *Export an MzTab object as mzTab file.*

Description

`writeMzTabData` exports an MzTab object as mzTab file. Note that the comment section "COM" are not written out.

Usage

```
writeMzTabData(
  object,
  file,
  what = c("MT", "PEP", "PRT", "PSM", "SML", "SMF", "SME")
)
```

Arguments

`object`	MzTab object, either read in by `MzTab()` or assembled.
`file`	`character(1)` with the file name.
`what`	`character` with names of the sections to be written out. Expected sections are `"MT"`, `"PEP"`, `"PRT"`, `"PSM"`, `"SML"`, `"SMF"`, or `"SME"`.

Author(s)

Steffen Neumann

Index

∗ **MSnExp**
MSnExp-class, 85
∗ **OnDiskMSnExp**
OnDiskMSnExp-class, 114
∗ **ProcessingStep**
ProcessingStep-class, 132
∗ **classes**
FeatComp-class, 47
FeaturesOfInterest-class, 49
MIAPE-class, 78
MSmap-class, 82
MSnExp-class, 85
MSnProcess-class, 89
MSnSet-class, 90
MSnSetList-class, 96
MzTab-class, 105
OnDiskMSnExp-class, 114
ProcessingStep-class, 132
pSet-class, 133
ReporterIons-class, 157
Spectrum-class, 162
Spectrum1-class, 165
Spectrum2-class, 165
∗ **datasets**
iTRAQ4, 62
itraqdata, 63
TMT6, 167
∗ **documentation, internal**
missing-data, 80
∗ **file**
readMgfData, 143
readMSnSet, 146
writeMgfData-methods, 169
∗ **internal**
Deprecated, 39
∗ **manip**
readMSnSet, 146
∗ **methods**
addIdentificationData-methods, 5
bin-methods, 11
calculateFragments-methods, 12
clean-methods, 24
compareSpectra-methods, 36
estimateNoise-methods, 42
extractPrecSpectra-methods, 44
normalise-methods, 109
pickPeaks-methods, 121
plot-methods, 123
plot.Spectrum.Spectrum-methods, 125
plot2d-methods, 127
plotDensity-methods, 128
plotMzDelta-methods, 129
plotNA-methods, 130
purityCorrect-methods, 137
quantify-methods, 140
removeNoId-methods, 154
removePeaks-methods, 155
removeReporters-methods, 156
smooth-methods, 161
trimMz-methods, 168
updateObject-methods, 169
writeMgfData-methods, 169
∗ **spectra combination functions**
consensusSpectrum, 37
meanMzInts, 75
[,FoICollection,ANY,ANY,ANY-method *(*FeaturesOfInterest-class*)*, 49
[,FoICollection,ANY,ANY-method *(*FeaturesOfInterest-class*)*, 49
[,FoICollection-method *(*FeaturesOfInterest-class*)*, 49
[,MChromatograms,ANY,ANY,ANY-method *(*MChromatograms*)*, 67
[,MSnSet,ANY,ANY,ANY-method *(*MSnSet-class*)*, 90
[,MSnSet,ANY,ANY-method *(*MSnSet-class*)*, 90

[,MSnSet-method *(MSnSet-class)*, 90
[,MSnSetList,ANY,ANY,ANY-method *(MSnSetList-class)*, 96
[,MSnSetList,ANY,missing,missing-method *(MSnSetList-class)*, 96
[,OnDiskMSnExp,ANY,ANY,ANY-method *(OnDiskMSnExp-class)*, 114
[,OnDiskMSnExp,logicalOrNumeric,missing,missing-method *(OnDiskMSnExp-class)*, 114
[,ReporterIons,ANY,ANY,ANY-method *(ReporterIons-class)*, 157
[,ReporterIons,ANY,ANY-method *(ReporterIons-class)*, 157
[,ReporterIons-method *(ReporterIons-class)*, 157
[,pSet,ANY,ANY,ANY-method *(pSet-class)*, 133
[,pSet,ANY,ANY-method *(pSet-class)*, 133
[,pSet-method *(pSet-class)*, 133
[<-,MChromatograms,ANY,ANY,ANY-method *(MChromatograms)*, 67
[[,FoICollection,ANY,ANY-method *(FeaturesOfInterest-class)*, 49
[[,FoICollection-method *(FeaturesOfInterest-class)*, 49
[[,MSnSetList,ANY,ANY-method *(MSnSetList-class)*, 96
[[,MSnSetList,ANY,missing-method *(MSnSetList-class)*, 96
[[,OnDiskMSnExp,ANY,ANY,missing-method *(OnDiskMSnExp-class)*, 114
[[,OnDiskMSnExp,ANY,ANY-method *(OnDiskMSnExp-class)*, 114
[[,OnDiskMSnExp,ANY,missing,missing-method *(OnDiskMSnExp-class)*, 114
[[,OnDiskMSnExp-method *(OnDiskMSnExp-class)*, 114
[[,pSet,ANY,ANY-method *(pSet-class)*, 133
[[,pSet-method *(pSet-class)*, 133
$,MChromatograms-method *(MChromatograms)*, 67
$,pSet-method *(pSet-class)*, 133
$<-,MChromatograms-method *(MChromatograms)*, 67
$<-,pSet-method *(pSet-class)*, 133

abstract,MIAPE-method *(MIAPE-class)*, 78
abstract,pSet-method *(pSet-class)*, 133
acquisitionNum *(Spectrum-class)*, 162
acquisitionNum,MSnSet-method *(MSnSet-class)*, 90
acquisitionNum,MSpectra-method *(MSpectra)*, 99
acquisitionNum,OnDiskMSnExp-method *(OnDiskMSnExp-class)*, 114
acquisitionNum,pSet-method *(pSet-class)*, 133
acquisitionNum,Spectrum-method *(Spectrum-class)*, 162
addFeaturesOfInterest *(FeaturesOfInterest-class)*, 49
addFeaturesOfInterest,FeaturesOfInterest,FoICollection-met *(FeaturesOfInterest-class)*, 49
addFeaturesOfInterest-methods *(FeaturesOfInterest-class)*, 49
addIdentificationData, *87*, *94*, *141*
addIdentificationData *(addIdentificationData-methods)*, 5
addIdentificationData,MSnExp,character-method *(MSnExp-class)*, 85
addIdentificationData,MSnExp,data.frame-method *(MSnExp-class)*, 85
addIdentificationData,MSnExp,mzID-method *(MSnExp-class)*, 85
addIdentificationData,MSnExp,mzIDClasses-method *(MSnExp-class)*, 85
addIdentificationData,MSnExp,mzIDCollection-method *(MSnExp-class)*, 85
addIdentificationData,MSnExp,mzRident-method *(MSnExp-class)*, 85
addIdentificationData,MSnSet,character-method *(MSnSet-class)*, 90
addIdentificationData,MSnSet,data.frame-method *(MSnSet-class)*, 90
addIdentificationData,MSnSet,mzID-method *(MSnSet-class)*, 90
addIdentificationData,MSnSet,mzIDClasses-method *(MSnSet-class)*, 90
addIdentificationData,MSnSet,mzIDCollection-method *(MSnSet-class)*, 90
addIdentificationData,MSnSet,mzRident-method *(MSnSet-class)*, 90
addIdentificationData-methods, 5
addMSnSetMetadata *(MSnSet-class)*, 90
aggregationFun *(Chromatogram)*, 14
aggvar, 7, *28*

alignRt,Chromatogram,Chromatogram-method (Chromatogram), 14
alignRt,MChromatograms,Chromatogram-method (MChromatograms), 67
all.equal,MSnExp,MSnExp-method (MSnExp-class), 85
all.equal,MSnExp,OnDiskMSnExp-method (MSnExp-class), 85
all.equal,OnDiskMSnExp,MSnExp-method (MSnExp-class), 85
all.equal,OnDiskMSnExp,OnDiskMSnExp-method (MSnExp-class), 85
analyser (MIAPE-class), 78
analyser,MIAPE-method (MIAPE-class), 78
analyser,MSnSet-method (MSnSet-class), 90
analyser,pSet-method (pSet-class), 133
analyserDetails (MIAPE-class), 78
analyserDetails,MIAPE-method (MIAPE-class), 78
analyserDetails,pSet-method (pSet-class), 133
analyzer (MIAPE-class), 78
analyzer,MIAPE-method (MIAPE-class), 78
analyzer,MSnSet-method (MSnSet-class), 90
analyzer,pSet-method (pSet-class), 133
analyzerDetails (MIAPE-class), 78
analyzerDetails,MIAPE-method (MIAPE-class), 78
analyzerDetails,pSet-method (pSet-class), 133
AnnotatedDataFrame, *86*, *90*, *91*, *96*, *114*, *115*, *134*, *143*, *145*
as, 8
as.data.frame,Chromatogram-method (Chromatogram), 14
as.data.frame.MSnExp (MSnExp-class), 85
as.data.frame.MSnSet (MSnSet-class), 90
as.data.frame.Spectrum (Spectrum-class), 162
as.ExpressionSet.MSnSet (MSnSet-class), 90
as.matrix.FoICollection (FeaturesOfInterest-class), 49
as.MIAME.MIAPE (MIAPE-class), 78
as.MSnExp.OnDiskMSnExp (OnDiskMSnExp-class), 114
as.MSnSet.ExpressionSet (MSnSet-class), 90
AssayData, *91*
assayData, *91*
assayData,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
assayData,pSet-method (pSet-class), 133
averageMSnSet, 9, *48*

bin, *36*, *86*, *163*
bin (bin-methods), 11
bin,Chromatogram-method (Chromatogram), 14
bin,MChromatograms-method (MChromatograms), 67
bin,MSnExp-method (MSnExp-class), 85
bin,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
bin,Spectrum-method (Spectrum-class), 162
bin-methods, 11
bpi (OnDiskMSnExp-class), 114
bpi,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
bpparam(), *30*

c,MChromatograms-method (MChromatograms), 67
calculateFragments, *125*, *126*, *166*
calculateFragments (calculateFragments-methods), 12
calculateFragments,character,missing-method (calculateFragments-methods), 12
calculateFragments,character,Spectrum2-method (Spectrum2-class), 165
calculateFragments-methods, 12
centroided (Spectrum-class), 162
centroided,MSpectra-method (MSpectra), 99
centroided,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
centroided,pSet-method (pSet-class), 133
centroided,Spectrum-method (Spectrum-class), 162
centroided<- (Spectrum-class), 162
centroided<-,OnDiskMSnExp,logical-method (OnDiskMSnExp-class), 114

centroided<-,pSet,ANY-method (pSet-class), 133
centroided<-,pSet,logical-method (pSet-class), 133
centroided<-,Spectrum,ANY-method (Spectrum-class), 162
centroided<-,Spectrum,logical-method (Spectrum-class), 162
Chromatogram, 14, *23–25*, *125*
chromatogram, *89*
chromatogram (chromatogram,MSnExp-method), 22
Chromatogram(), *18*, *67*, *69*, *71–73*
chromatogram(), *14*, *71*
chromatogram,MSnExp-method, 22
Chromatogram-class (Chromatogram), 14
class:MIAPE (MIAPE-class), 78
class:MSnExp (MSnExp-class), 85
class:MSnProcess (MSnProcess-class), 89
class:MSnSet (MSnSet-class), 90
class:MzTab (MzTab-class), 105
class:NAnnotatedDataFrame (Deprecated), 39
class:OnDiskMSnExp (OnDiskMSnExp-class), 114
class:pSet (pSet-class), 133
class:ReporterIons (ReporterIons-class), 157
class:Spectrum (Spectrum-class), 162
class:Spectrum1 (Spectrum1-class), 165
class:Spectrum2 (Spectrum2-class), 165
clean, *11*, *36*, *86*, *89*, *114*, *118*, *122*, *155–157*, *161*, *163*, *168*
clean (clean-methods), 24
clean(), *17*, *19*, *73*, *103*
clean,Chromatogram-method (Chromatogram), 14
clean,MChromatograms-method (MChromatograms), 67
clean,MSnExp-method (MSnExp-class), 85
clean,MSpectra-method (MSpectra), 99
clean,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
clean,Spectrum-method (Spectrum-class), 162
clean-methods, 24
closest(), *19*
coerce,AnnotatedDataFrame,list-method (pSet-class), 133
coerce,ExpressionSet,MSnSet-method (MSnSet-class), 90
coerce,FoICollection,matrix-method (FeaturesOfInterest-class), 49
coerce,IBSpectra,MSnSet-method (MSnSet-class), 90
coerce,matrix,MChromatograms-method (MChromatograms), 67
coerce,MIAPE,MIAME-method (MIAPE-class), 78
coerce,MIAxE,list-method (pSet-class), 133
coerce,MSmap,data.frame-method (MSmap-class), 82
coerce,MSnExp,data.frame-method (MSnExp-class), 85
coerce,MSnExp,MSpectra-method (MSnExp-class), 85
coerce,MSnProcess,list-method (MSnProcess-class), 89
coerce,MSnSet,data.frame-method (MSnSet-class), 90
coerce,MSnSet,ExpressionSet-method (MSnSet-class), 90
coerce,MSnSet,SummarizedExperiment-method (MSnSet-class), 90
coerce,MSpectra,list-method (MSpectra), 99
coerce,MSpectra,MSnExp-method (MSpectra), 99
coerce,mzRident,data.frame-method (readMzIdData), 148
coerce,MzTab,MSnSetList-method (MzTab-class), 105
coerce,OnDiskMSnExp,MSnExp-method (OnDiskMSnExp-class), 114
coerce,Spectra,MSpectra-method (extractSpectraData), 45
coerce,Spectrum,data.frame-method (Spectrum-class), 162
coerce,SummarizedExperiment,MSnSet-method (MSnSet-class), 90
collisionEnergy (Spectrum2-class), 165
collisionEnergy,MSpectra-method (MSpectra), 99
collisionEnergy,OnDiskMSnExp-method

(OnDiskMSnExp-class), 114
collisionEnergy,pSet-method (pSet-class), 133
collisionEnergy,Spectrum-method (Spectrum2-class), 165
colnames<-,MChromatograms,ANY-method (MChromatograms), 67
combine,MIAPE,MIAPE-method (MIAPE-class), 78
combine,MSnProcess,MSnProcess-method (MSnProcess-class), 89
combine,MSnSet,MSnSet-method (MSnSet-class), 90
combineFeatures, *8*, 26, *59*, *111*, *113*, *140*
combineFeatures(), *49*
combineFeatures,MSnSet-method (combineFeatures), 26
combineSpectra (combineSpectra,MSnExp-method), 29
combineSpectra,MSnExp-method, 29
combineSpectra,MSpectra-method (combineSpectra,MSnExp-method), 29
combineSpectraMovingWindow, 32
combineSpectraMovingWindow(), *41*, *77*
comments (MzTab-class), 105
common (FeatComp-class), 47
common,FeatComp-method (FeatComp-class), 47
common,methods (FeatComp-class), 47
commonFeatureNames, 34, *96*, *98*
compareChromatograms (Chromatogram), 14
compareChromatograms,Chromatogram,Chromatogram-method (Chromatogram), 14
compareChromatograms,MChromatograms,MChromatograms-method (MChromatograms), 67
compareChromatograms,MChromatograms,missing-method (MChromatograms), 67
compareMSnSets, 35
compareSpectra, *86*, *164*
compareSpectra (compareSpectra-methods), 36
compareSpectra,MSnExp,missing-method (MSnExp-class), 85
compareSpectra,OnDiskMSnExp,missing-method (OnDiskMSnExp-class), 114
compareSpectra,Spectrum,Spectrum-method (Spectrum-class), 162
compareSpectra-methods, 36
compfnames, *10*
compfnames (FeatComp-class), 47
compfnames,list,missing-method (FeatComp-class), 47
compfnames,MSnSet,MSnSet-method (FeatComp-class), 47
compfnames-methods (FeatComp-class), 47
consensusSpectrum, 37
consensusSpectrum(), *77*

data, *157*
DataFrame, *101*
DataFrame(), *45*
Deprecated, 39
description,FeaturesOfInterest-method (FeaturesOfInterest-class), 49
description,FoICollection-method (FeaturesOfInterest-class), 49
description,MSnSet-method (MSnSet-class), 90
description,pSet-method (pSet-class), 133
description,ReporterIons-method (ReporterIons-class), 157
detectorType (MIAPE-class), 78
detectorType,MIAPE-method (MIAPE-class), 78
detectorType,MSnSet-method (MSnSet-class), 90
detectorType,pSet-method (pSet-class), 133
dim (pSet-class), 133
dim,MSmap-method (MSmap-class), 82
dim,MSnSet-method (MSnSet-class), 90
dim,NAnnotatedDataFrame-method (Deprecated), 39
dim,pSet-method (pSet-class), 133
droplevels, *93*
droplevels.MSnSet (MSnSet-class), 90

eSet, *86*, *90*, *91*, *95*, *115*, *133*, *134*, *136*
estimateMzResolution (estimateMzResolution,MSnExp-method), 39
estimateMzResolution(), *41*, *77*
estimateMzResolution,MSnExp-method, 39

estimateMzResolution,Spectrum-method (estimateMzResolution,MSnExp-method), 39
estimateMzScattering, 41
estimateMzScattering(), *33*, *76, 77*
estimateNoise, *86*, *122*, *164*
estimateNoise (estimateNoise-methods), 42
estimateNoise,MSnExp-method (MSnExp-class), 85
estimateNoise,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
estimateNoise,Spectrum-method (Spectrum-class), 162
estimateNoise-methods, 42
executeProcessingStep (ProcessingStep-class), 132
expandFeatureVars, 43
expemail (MIAPE-class), 78
expemail,MIAPE-method (MIAPE-class), 78
expemail,MSnSet-method (MSnSet-class), 90
expemail,pSet-method (pSet-class), 133
experimentData, *86*, *91*, *115*, *134*
experimentData,pSet-method (pSet-class), 133
experimentData<-,MSnSet,MIAPE-method (MSnSet-class), 90
expinfo,MIAPE-method (MIAPE-class), 78
ExpressionSet, *90*, *95*
exprs, *91*, *110*
exprs,MSnSet-method (MSnSet-class), 90
exptitle (MIAPE-class), 78
exptitle,MIAPE-method (MIAPE-class), 78
exptitle,MSnSet-method (MSnSet-class), 90
exptitle,pSet-method (pSet-class), 133
extractPrecSpectra, *86*, *123*
extractPrecSpectra (extractPrecSpectra-methods), 44
extractPrecSpectra,MSnExp,numeric-method (MSnExp-class), 85
extractPrecSpectra,MSnExp-method (MSnExp-class), 85
extractPrecSpectra,OnDiskMSnExp,numeric-method (OnDiskMSnExp-class), 114
extractPrecSpectra-methods, 44
extractSpectra (Deprecated), 39
extractSpectraData, 45
extractSpectraData(), *103*

factorsAsStrings, 46
fData,MChromatograms-method (MChromatograms), 67
fData,MSnSetList-method (MSnSetList-class), 96
fData,pSet-method (pSet-class), 133
fData<-,MChromatograms,ANY-method (MChromatograms), 67
fData<-,MSnSet,data.frame-method (MSnSet-class), 90
fData<-,MSnSetList,DataFrame-method (MSnSetList-class), 96
fData<-,pSet,data.frame-method (pSet-class), 133
FeatComp-class, 47
featureCV, *27*, *28*, 48
featureData, *86*, *91*, *114*, *134*
featureData,MChromatograms-method (MChromatograms), 67
featureData,pSet-method (pSet-class), 133
featureData<-,MChromatograms,ANY-method (MChromatograms), 67
featureNames,MChromatograms-method (MChromatograms), 67
featureNames,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
featureNames,pSet-method (pSet-class), 133
featureNames<-,MChromatograms-method (MChromatograms), 67
featureNames<-,OnDiskMSnExp,ANY-method (OnDiskMSnExp-class), 114
featureNames<-,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
FeaturesOfInterest (FeaturesOfInterest-class), 49
FeaturesOfInterest,character,character,missing-method (FeaturesOfInterest-class), 49
FeaturesOfInterest,character,character,MSnSet-method (FeaturesOfInterest-class), 49
FeaturesOfInterest-class, 49
FeaturesOfInterest-methods (FeaturesOfInterest-class), 49
fileName,MSmap-method (MSmap-class), 82

fileName,MzTab-method (MzTab-class), 105
fileNames (pSet-class), 133
fileNames,MSmap-method (MSmap-class), 82
fileNames,MSnProcess-method (MSnProcess-class), 89
fileNames,MSnSet-method (MSnSet-class), 90
fileNames,MzTab-method (MzTab-class), 105
fileNames,pSet-method (pSet-class), 133
fillUp, 52
filterAcquisitionNum (MSnExp-class), 85
filterAcquisitionNum,MSnExp-method (MSnExp-class), 85
filterAcquisitionNum,OnDiskMSnExp-method (MSnExp-class), 85
filterEmptySpectra (MSnExp-class), 85
filterEmptySpectra,MSnExp-method (MSnExp-class), 85
filterEmptySpectra,OnDiskMSnExp-method (MSnExp-class), 85
filterFile (MSnExp-class), 85
filterFile,MSnExp-method (MSnExp-class), 85
filterFile,OnDiskMSnExp-method (MSnExp-class), 85
filterIdentificationDataFrame, *5–7*, 53
filterIdentificationDataFrame(), *149*
filterIntensity,Chromatogram-method (Chromatogram), 14
filterIntensity,MChromatograms-method (MChromatograms), 67
filterIsolationWindow (MSnExp-class), 85
filterIsolationWindow,MSnExp-method (MSnExp-class), 85
filterMsLevel (MSnExp-class), 85
filterMsLevel,MSnExp-method (MSnExp-class), 85
filterMsLevel,MSnSet-method (MSnSet-class), 90
filterMsLevel,MSpectra-method (MSpectra), 99
filterMsLevel,OnDiskMSnExp-method (MSnExp-class), 85
filterMz, *88*, *124*
filterMz (trimMz-methods), 168
filterMz(), *101*, *103*
filterMz,MSnExp-method (MSnExp-class), 85
filterMz,MSpectra-method (MSpectra), 99
filterMz,OnDiskMSnExp-method (MSnExp-class), 85
filtermz,Spectrum,numeric-method (Spectrum-class), 162
filterMz,Spectrum-method (Spectrum-class), 162
filterMz-methods (trimMz-methods), 168
filterNA, *27*, *131*
filterNA (MSnSet-class), 90
filterNA,matrix-method (MSnSet-class), 90
filterNA,MSnSet-method (MSnSet-class), 90
filterPolarity (MSnExp-class), 85
filterPolarity,MSnExp-method (MSnExp-class), 85
filterPolarity,OnDiskMSnExp-method (MSnExp-class), 85
filterPrecursorMz (MSnExp-class), 85
filterPrecursorMz,MSnExp-method (MSnExp-class), 85
filterPrecursorScan (MSnExp-class), 85
filterPrecursorScan,MSnExp-method (MSnExp-class), 85
filterPrecursorScan,OnDiskMSnExp-method (MSnExp-class), 85
filterRt, *124*
filterRt (MSnExp-class), 85
filterRt,Chromatogram-method (Chromatogram), 14
filterRt,MSnExp-method (MSnExp-class), 85
filterRt,OnDiskMSnExp-method (MSnExp-class), 85
filterZero (MSnSet-class), 90
filterZero,matrix-method (MSnSet-class), 90
filterZero,MSnSet-method (MSnSet-class), 90
fnamesIn (FeaturesOfInterest-class), 49
fnamesIn,FeaturesOfInterest,data.frame-method (FeaturesOfInterest-class), 49
fnamesIn,FeaturesOfInterest,matrix-method (FeaturesOfInterest-class), 49
fnamesIn,FeaturesOfInterest,MSnSet-method (FeaturesOfInterest-class), 49

fnamesIn-methods
(FeaturesOfInterest-class), 49
foi (FeaturesOfInterest-class), 49
foi,FeaturesOfInterest-method
(FeaturesOfInterest-class), 49
foi,FoICollection-method
(FeaturesOfInterest-class), 49
foi-methods (FeaturesOfInterest-class),
49
FoICollection
(FeaturesOfInterest-class), 49
FoICollection,list-method
(FeaturesOfInterest-class), 49
FoICollection,missing-method
(FeaturesOfInterest-class), 49
FoICollection-class
(FeaturesOfInterest-class), 49
FoICollection-methods
(FeaturesOfInterest-class), 49
formatRt, 54
fromFile (Spectrum-class), 162
fromFile,Chromatogram-method
(Chromatogram), 14
fromFile,MSnSet-method (MSnSet-class),
90
fromFile,MSpectra-method (MSpectra), 99
fromFile,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
fromFile,pSet-method (pSet-class), 133
fromFile,Spectrum-method
(Spectrum-class), 162
fvarLabels,MChromatograms-method
(MChromatograms), 67
fvarLabels,pSet-method (pSet-class), 133
fvarMetadata,pSet-method (pSet-class),
133

geom_histogram, *129*
getEcols, *148*
getEcols (grepEcols), 55
getVariableName, 54
grep, *55*
grepEcols, 55, *148*

hasChromatograms (hasSpectra), 56
hasSpectra, 56
header (pSet-class), 133
header,OnDiskMSnExp,missing-method
(OnDiskMSnExp-class), 114
header,OnDiskMSnExp,numeric-method
(OnDiskMSnExp-class), 114
header,pSet,missing-method
(pSet-class), 133
header,pSet,numeric-method
(pSet-class), 133
hist, *11*

idSummary (MSnSet-class), 90
idSummary,MSnExp-method (MSnExp-class),
85
idSummary,MSnSet-method (MSnSet-class),
90
image, *57*
image,MSnSet-method (MSnSet-class), 90
image2 (MSnSet-class), 90
imageNA2, 57, *80*, *81*
impute, *27*, *93*
impute,MSnSet-method, 58
instrumentCustomisations (MIAPE-class),
78
instrumentCustomisations,MIAPE-method
(MIAPE-class), 78
instrumentCustomisations,pSet-method
(pSet-class), 133
instrumentManufacturer (MIAPE-class), 78
instrumentManufacturer,MIAPE-method
(MIAPE-class), 78
instrumentManufacturer,pSet-method
(pSet-class), 133
instrumentModel (MIAPE-class), 78
instrumentModel,MIAPE-method
(MIAPE-class), 78
instrumentModel,pSet-method
(pSet-class), 133
intensity (Spectrum-class), 162
intensity,Chromatogram-method
(Chromatogram), 14
intensity,MSpectra-method (MSpectra), 99
intensity,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
intensity,pSet-method (pSet-class), 133
intensity,Spectrum-method
(Spectrum-class), 162
ionCount (Spectrum-class), 162
ionCount,MSpectra-method (MSpectra), 99
ionCount,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
ionCount,pSet-method (pSet-class), 133

ionCount,Spectrum-method
(Spectrum-class), 162
ionSource (MIAPE-class), 78
ionSource,MIAPE-method (MIAPE-class), 78
ionSource,MSnSet-method (MSnSet-class), 90
ionSource,pSet-method (pSet-class), 133
ionSourceDetails (MIAPE-class), 78
ionSourceDetails,MIAPE-method
(MIAPE-class), 78
ionSourceDetails,pSet-method
(pSet-class), 133
iPQF, *26*, *28*, 59, *63*
is.na.MSnSet, *93*
is.na.MSnSet (plotNA-methods), 130
isCentroided (Spectrum-class), 162
isCentroided(), *61*
isCentroided,MSnExp-method
(MSnExp-class), 85
isCentroided,MSpectra-method
(MSpectra), 99
isCentroided,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
isCentroided,Spectrum-method
(Spectrum-class), 162
isCentroidedFromFile, 61, *116*
isEmpty,Chromatogram-method
(Chromatogram), 14
isEmpty,environment-method
(Spectrum-class), 162
isEmpty,MChromatograms-method
(MChromatograms), 67
isEmpty,MSpectra-method (MSpectra), 99
isEmpty,Spectrum-method
(Spectrum-class), 162
isMSnbaseFastLoad (MSnbaseOptions), 84
isMSnbaseVerbose, *6*
isMSnbaseVerbose (MSnbaseOptions), 84
isMSnbaseVerbose(), *145*
isolationWindow, *87*
isolationWindow,MSnExp-method
(MSnExp-class), 85
isolationWindowLowerMz (pSet-class), 133
isolationWindowLowerMz,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
isolationWindowLowerMz,pSet-method
(pSet-class), 133
isolationWindowUpperMz (pSet-class), 133
isolationWindowUpperMz,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
isolationWindowUpperMz,pSet-method
(pSet-class), 133
iTRAQ4, 62, *159*, *167*
iTRAQ5 (iTRAQ4), 62
iTRAQ8 (iTRAQ4), 62
iTRAQ9 (iTRAQ4), 62
itraqdata, 63

lapply,MSnSetList-method
(MSnSetList-class), 96
length (pSet-class), 133
length,Chromatogram-method
(Chromatogram), 14
length,FeaturesOfInterest-method
(FeaturesOfInterest-class), 49
length,FoICollection-method
(FeaturesOfInterest-class), 49
length,MSnSetList-method
(MSnSetList-class), 96
length,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
length,pSet-method (pSet-class), 133
length,ReporterIons-method
(ReporterIons-class), 157
length-method (ReporterIons-class), 157
lengths,FoICollection-method
(FeaturesOfInterest-class), 49
listOf, 64
log,MSnSet-method (MSnSet-class), 90

ma.plot, *94*
mad, *112*
makeCamelCase, 64
makeImpuritiesMatrix
(purityCorrect-methods), 137
makeMTD (Deprecated), 39
makeNaData, 65
makeNaData2 (makeNaData), 65
makePEP (Deprecated), 39
makePRT (Deprecated), 39
MAplot,MSnSet-method (MSnSet-class), 90
MChromatograms, *20*, *23*, 67
MChromatograms(), *19*, *151*
MChromatograms-class (MChromatograms), 67
mcols(), *99*
meanMzInts, 75

meanMzInts(), *30*, *32*, *33*, *37*, *38*
meanSdPlot, *93*
meanSdPlot,MSnSet-method *(*MSnSet-class*)*, 90
mergeFeatureVars *(*expandFeatureVars*)*, 43
metadata,MzTab-method *(*MzTab-class*)*, 105
MIAME, *79*
MIAPE, *86*, *91*, *93*, *115*, *134*, *136*
MIAPE *(*MIAPE-class*)*, 78
MIAPE-class, 78
MIAxE, *80*
missing-data, 80
missingdata *(*missing-data*)*, 80
moleculeEvidence *(*MzTab-class*)*, 105
moleculeFeatures *(*MzTab-class*)*, 105
ms2df *(*MSnSet-class*)*, 90
MsCoreUtils::formatRt(), *54*
MsCoreUtils::impute_matrix(), *58*
msInfo *(*MIAPE-class*)*, 78
msInfo,MIAPE-method *(*MIAPE-class*)*, 78
msInfo,MSnSet-method *(*MSnSet-class*)*, 90
msInfo,pSet-method *(*pSet-class*)*, 133
msLevel *(*Spectrum-class*)*, 162
msLevel,Chromatogram-method *(*Chromatogram*)*, 14
msLevel,MSmap-method *(*MSmap-class*)*, 82
msLevel,MSpectra-method *(*MSpectra*)*, 99
msLevel,OnDiskMSnExp-method *(*OnDiskMSnExp-class*)*, 114
msLevel,pSet-method *(*pSet-class*)*, 133
msLevel,Spectrum-method *(*Spectrum-class*)*, 162
MSmap *(*MSmap-class*)*, 82
msMap *(*MSmap-class*)*, 82
msMap,MSmap-method *(*MSmap-class*)*, 82
MSmap,mzRpwiz-method *(*MSmap-class*)*, 82
MSmap,mzRramp-method *(*MSmap-class*)*, 82
MSmap,mzRraw-method *(*MSmap-class*)*, 82
MSmap,OnDiskMSnExp-method *(*MSmap-class*)*, 82
MSmap-class, 82
MSmap-method *(*MSmap-class*)*, 82
MSnbase-defunct *(*Deprecated*)*, 39
MSnbase-deprecated *(*Deprecated*)*, 39
MSnbaseOptions, 84
MSnExp, *5*, *14*, *22*, *29*, *39*, *44*, *45*, *90*, *91*, *109*, *114*, *115*, *118*, *119*, *123*, *127*, *128*, *132*, *133*, *136*, *140*, *143–145*, *154*, *156*, *157*, *163*, *165*, *166*, *169*, *170*
MSnExp *(*MSnExp-class*)*, 85
MSnExp-class, 85
MSnProcess, *86*, *91*, *115*, *134*, *143*
MSnProcess *(*MSnProcess-class*)*, 89
MSnProcess-class, 89
MSnSet, *5*, *6*, *26*, *27*, *34*, *35*, *48*, *49*, *65*, *90*, *96*, *97*, *109*, *110*, *113*, *137*, *141*, *146*, *148–151*, *154*, *163*
MSnSet *(*MSnSet-class*)*, 90
msnset *(*itraqdata*)*, 63
MSnSet-class, 90
msnset2 *(*itraqdata*)*, 63
MSnSetList, *106*, *150*
MSnSetList *(*MSnSetList-class*)*, 96
MSnSetList-class, 96
msnsets *(*MSnSetList-class*)*, 96
MSpectra, *29*, *45*, *88*, 99, *102*
MSpectra-class *(*MSpectra*)*, 99
multiLabels *(*Deprecated*)*, 39
multiLabels,NAnnotatedDataFrame-method *(*Deprecated*)*, 39
multiplex *(*Deprecated*)*, 39
multiplex,NAnnotatedDataFrame-method *(*Deprecated*)*, 39
mva.pairs, *94*
mz *(*Spectrum-class*)*, 162
mz,Chromatogram-method *(*Chromatogram*)*, 14
mz,MChromatograms-method *(*MChromatograms*)*, 67
mz,MSmap-method *(*MSmap-class*)*, 82
mz,MSpectra-method *(*MSpectra*)*, 99
mz,OnDiskMSnExp-method *(*OnDiskMSnExp-class*)*, 114
mz,pSet-method *(*pSet-class*)*, 133
mz,ReporterIons-method *(*ReporterIons-class*)*, 157
mz,Spectrum-method *(*Spectrum-class*)*, 162
mzR::writeMSData(), *171*
mzRes *(*MSmap-class*)*, 82
mzRes,MSmap-method *(*MSmap-class*)*, 82
MzTab, *97*, *150*, *172*
MzTab *(*MzTab-class*)*, 105
MzTab-class, 105
mzTabMode *(*MzTab-class*)*, 105
mzTabType *(*MzTab-class*)*, 105

names,FeatComp-method *(*FeatComp-class*)*,

47
names,FoICollection-method (FeaturesOfInterest-class), 49
names,MSnSetList-method (MSnSetList-class), 96
names,ReporterIons-method (ReporterIons-class), 157
names<-,FoICollection,character-method (FeaturesOfInterest-class), 49
names<-,MSnSetList,ANY-method (MSnSetList-class), 96
NAnnotatedDataFrame (Deprecated), 39
NAnnotatedDataFrame-class (Deprecated), 39
naplot, 107
naset (impute,MSnSet-method), 58
ncol,MSmap-method (MSmap-class), 82
nFeatures, *28*, 108
normalise, *92*
normalise (normalise-methods), 109
normalise(), *48*
normalise,MSnExp-method (normalise-methods), 109
normalise,MSnSet-method (normalise-methods), 109
normalise,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
normalise,Spectrum-method (normalise-methods), 109
normalise,Spectrum2-method (normalise-methods), 109
normalise-methods, 109
normalize, *118*
normalize (normalise-methods), 109
normalize,Chromatogram-method (Chromatogram), 14
normalize,MChromatograms-method (MChromatograms), 67
normalize,MSnExp-method (normalise-methods), 109
normalize,MSnSet-method (normalise-methods), 109
normalize,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
normalize,Spectrum-method (normalise-methods), 109
normalize,Spectrum2-method (normalise-methods), 109
normalize-methods (normalise-methods), 109
normToReference, 110
notes,MIAPE-method (MIAPE-class), 78
notes,pSet-method (pSet-class), 133
notes<-,MIAPE-method (MIAPE-class), 78
npcv, *9*, 112
nQuants, *93*, 113
nrow,MSmap-method (MSmap-class), 82
NTR, *26*, *28*
NTR (normToReference), 110

objlog (MSnSetList-class), 96
OnDiskMSnExp, *14*, *22*, *29*, *39*, *45*, *61*, *82*, *84*, *88*, *89*, *123*, *132*, *133*, *136*, *145*, *170*
OnDiskMSnExp (OnDiskMSnExp-class), 114
OnDiskMSnExp-class, 114
otherInfo,MIAPE-method (MIAPE-class), 78

par, *126*
pData,MChromatograms-method (MChromatograms), 67
pData,pSet-method (pSet-class), 133
pData<-,MChromatograms,data.frame-method (MChromatograms), 67
pData<-,MSnSet,data.frame-method (MSnSet-class), 90
pData<-,pSet,ANY-method (pSet-class), 133
peaksCount (Spectrum-class), 162
peaksCount,MSpectra,ANY-method (MSpectra), 99
peaksCount,OnDiskMSnExp,missing-method (OnDiskMSnExp-class), 114
peaksCount,OnDiskMSnExp,numeric-method (OnDiskMSnExp-class), 114
peaksCount,pSet,missing-method (pSet-class), 133
peaksCount,pSet,numeric-method (pSet-class), 133
peaksCount,Spectrum,missing-method (Spectrum-class), 162
peptides,MzTab-method (MzTab-class), 105
phenoData, *86*, *91*, *114*, *134*
phenoData,MChromatograms-method (MChromatograms), 67
phenoData,pSet-method (pSet-class), 133
phenoData<-,pSet,ANY-method (pSet-class), 133

pickPeaks, *11*, *36*, *43*, *86*, *161*, *164*
pickPeaks *(*pickPeaks-methods*)*, 121
pickPeaks(), *102*, *103*
pickPeaks,MSnExp-method *(*MSnExp-class*)*, 85
pickPeaks,MSpectra-method *(*MSpectra*)*, 99
pickPeaks,OnDiskMSnExp-method *(*OnDiskMSnExp-class*)*, 114
pickPeaks,Spectrum-method *(*Spectrum-class*)*, 162
pickPeaks-methods, 121
plot *(*plot-methods*)*, 123
plot,Chromatogram,ANY-method *(*Chromatogram*)*, 14
plot,MChromatograms,ANY-method *(*MChromatograms*)*, 67
plot,MSmap,missing-method *(*MSmap-class*)*, 82
plot,MSnExp *(*MSnExp-class*)*, 85
plot,MSnExp,missing-method *(*MSnExp-class*)*, 85
plot,Spectrum,missing-method *(*plot-methods*)*, 123
plot,Spectrum,Spectrum-method *(*plot.Spectrum.Spectrum-methods*)*, 125
plot,Spectrum-method *(*plot-methods*)*, 123
plot,Spectrum2,character-method *(*plot-methods*)*, 123
plot-methods, 123
plot.default, *126*
plot.MSnExp, *86*
plot.MSnExp *(*plot-methods*)*, 123
plot.Spectrum, *126*, *164*
plot.Spectrum *(*plot-methods*)*, 123
plot.Spectrum.character, *164*
plot.Spectrum.Spectrum, *125*, *164*
plot.Spectrum.Spectrum *(*plot.Spectrum.Spectrum-methods*)*, 125
plot.Spectrum.Spectrum-methods, 125
plot2d, *87*, *128*, *130*
plot2d *(*plot2d-methods*)*, 127
plot2d,data.frame-method *(*plot2d-methods*)*, 127
plot2d,MSnExp-method *(*plot2d-methods*)*, 127
plot2d-methods, 127
plot3D *(*MSmap-class*)*, 82
plot3D,MSmap-method *(*MSmap-class*)*, 82
plotDensity, *87*, *127*, *128*, *130*
plotDensity *(*plotDensity-methods*)*, 128
plotDensity,data.frame-method *(*plotDensity-methods*)*, 128
plotDensity,MSnExp-method *(*plotDensity-methods*)*, 128
plotDensity-methods, 128
plotMzDelta, *87*, *127*
plotMzDelta *(*plotMzDelta-methods*)*, 129
plotMzDelta,MSnExp-method *(*plotMzDelta-methods*)*, 129
plotMzDelta,mzRramp-method *(*plotMzDelta-methods*)*, 129
plotMzDelta-methods, 129
plotNA, *80*, *81*, *93*, *94*
plotNA *(*plotNA-methods*)*, 130
plotNA,matrix-method *(*plotNA-methods*)*, 130
plotNA,MSnSet-method *(*plotNA-methods*)*, 130
plotNA-methods, 130
polarity *(*Spectrum-class*)*, 162
polarity,MChromatograms-method *(*MChromatograms*)*, 67
polarity,MSpectra-method *(*MSpectra*)*, 99
polarity,OnDiskMSnExp-method *(*OnDiskMSnExp-class*)*, 114
polarity,pSet-method *(*pSet-class*)*, 133
polarity,Spectrum-method *(*Spectrum-class*)*, 162
precAcquisitionNum *(*Spectrum2-class*)*, 165
precAcquisitionNum,pSet-method *(*pSet-class*)*, 133
precAcquisitionNum,Spectrum-method *(*Spectrum2-class*)*, 165
precScanNum *(*Spectrum2-class*)*, 165
precScanNum,MSpectra-method *(*MSpectra*)*, 99
precScanNum,OnDiskMSnExp-method *(*OnDiskMSnExp-class*)*, 114
precScanNum,pSet-method *(*pSet-class*)*, 133
precScanNum,Spectrum-method *(*Spectrum2-class*)*, 165
precSelection, 131

precSelectionTable (precSelection), 131
precursorCharge (Spectrum2-class), 165
precursorCharge,MSpectra-method (MSpectra), 99
precursorCharge,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
precursorCharge,pSet-method (pSet-class), 133
precursorCharge,Spectrum-method (Spectrum2-class), 165
precursorIntensity (Spectrum2-class), 165
precursorIntensity,MSpectra-method (MSpectra), 99
precursorIntensity,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
precursorIntensity,pSet-method (pSet-class), 133
precursorIntensity,Spectrum-method (Spectrum2-class), 165
precursorMz, *129*
precursorMz (Spectrum2-class), 165
precursorMz,Chromatogram-method (Chromatogram), 14
precursorMz,MChromatograms-method (MChromatograms), 67
precursorMz,MSpectra-method (MSpectra), 99
precursorMz,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
precursorMz,pSet-method (pSet-class), 133
precursorMz,Spectrum-method (Spectrum2-class), 165
processingData (pSet-class), 133
processingData,MSnSet-method (MSnSet-class), 90
processingData,pSet-method (pSet-class), 133
ProcessingStep, *114*
ProcessingStep (ProcessingStep-class), 132
ProcessingStep-class, 132
ProcessingStep:OnDiskMSnExp (ProcessingStep-class), 132
productMz,Chromatogram-method (Chromatogram), 14
productMz,MChromatograms-method (MChromatograms), 67
proteins,MzTab-method (MzTab-class), 105
protocolData, *86*, *91*, *115*, *134*
protocolData,pSet-method (pSet-class), 133
pSet, *85*, *86*, *89*, *91*, *114*, *115*, *118*, *119*
pSet (pSet-class), 133
pSet-class, 133
psms,MzTab-method (MzTab-class), 105
pubMedIds,MIAPE-method (MIAPE-class), 78
pubMedIds,pSet-method (pSet-class), 133
pubMedIds<-,MIAPE-method (MIAPE-class), 78
purityCorrect, *92*
purityCorrect (purityCorrect-methods), 137
purityCorrect,MSnSet,matrix-method (MSnSet-class), 90
purityCorrect,MSnSet-method (MSnSet-class), 90
purityCorrect-methods, 137

qual (MSnSet-class), 90
qual,MSnSet-method (MSnSet-class), 90
quantify, *62*, *63*, *87*, *91*, *95*, *158*, *164*, *167*
quantify (quantify-methods), 140
quantify,MSnExp,character-method (MSnExp-class), 85
quantify,MSnExp-method (MSnExp-class), 85
quantify,OnDiskMSnExp-method (OnDiskMSnExp-class), 114
quantify,OnMSnExp-method (MSnExp-class), 85
quantify,Spectrum,character-method (Spectrum-class), 162
quantify,Spectrum-method (Spectrum-class), 162
quantify-methods, 140

read.AnnotatedDataFrame, *147*
read.csv, *147*
read.MIAME, *147*
read.table, *147*
readExpressionSet, *146*
readLines, *55*, *147*
readMgfData, 143, *170*
readMgfData(), *146*
readMSData, *85*, *89*, *114*, *119*, *144*, 144

readMSData(), *102*, *171*
readMSData2 *(*readMSData*)*, 144
readMSnSet, *90*, *95*, 146
readMSnSet2, *55*, *56*, *90*
readMSnSet2 *(*readMSnSet*)*, 146
readMzIdData, *7*, 148
readMzIdData(), *53*
readMzTabData, 149
readMzTabData_v0.9, *150*, 150
readMzXMLData *(*Deprecated*)*, 39
readSRMData, 151
reduce, *8*, *53*
reduce,data.frame-method, 153
removeMultipleAssignment, *141*
removeMultipleAssignment *(*MSnSet-class*)*, 90
removeMultipleAssignment,MSnExp-method *(*MSnExp-class*)*, 85
removeMultipleAssignment,MSnSet-method *(*MSnSet-class*)*, 90
removeMultipleAssignment-method *(*MSnSet-class*)*, 90
removeNoId, *87*, *94*
removeNoId *(*removeNoId-methods*)*, 154
removeNoId,MSnExp-method *(*MSnExp-class*)*, 85
removeNoId,MSnSet-method *(*MSnSet-class*)*, 90
removeNoId-methods, 154
removePeaks, *11*, *25*, *36*, *87*, *89*, *114*, *118*, *122*, *157*, *161*, *164*, *168*
removePeaks *(*removePeaks-methods*)*, 155
removePeaks(), *103*
removePeaks,MSnExp-method *(*MSnExp-class*)*, 85
removePeaks,MSpectra-method *(*MSpectra*)*, 99
removePeaks,OnDiskMSnExp-method *(*OnDiskMSnExp-class*)*, 114
removePeaks,Spectrum-method *(*Spectrum-class*)*, 162
removePeaks-methods, 155
removeReporters, *87*, *166*
removeReporters *(*removeReporters-methods*)*, 156
removeReporters,MSnExp-method *(*MSnExp-class*)*, 85
removeReporters,OnDiskMSnExp-method *(*MSnExp-class*)*, 85
removeReporters,Spectrum-method *(*Spectrum2-class*)*, 165
removeReporters-methods, 156
reporterColors *(*ReporterIons-class*)*, 157
reporterColors,ReporterIons-method *(*ReporterIons-class*)*, 157
reporterColors-method *(*ReporterIons-class*)*, 157
reporterColours *(*ReporterIons-class*)*, 157
reporterColours,ReporterIons-method *(*ReporterIons-class*)*, 157
reporterColours-method *(*ReporterIons-class*)*, 157
ReporterIons, *62*, *123*, *129*, *140*, *141*, *156*, *167*
ReporterIons *(*ReporterIons-class*)*, 157
ReporterIons-class, 157
reporterNames *(*ReporterIons-class*)*, 157
reporterNames,ReporterIons-method *(*ReporterIons-class*)*, 157
reporterNames-method *(*ReporterIons-class*)*, 157
reporterNames<- *(*ReporterIons-class*)*, 157
reporterNames<-,ReporterIons,ANY-method *(*ReporterIons-class*)*, 157
reporterNames<-,ReporterIons,character-method *(*ReporterIons-class*)*, 157
reporterNames<-,ReporterIons-method *(*ReporterIons-class*)*, 157
requiredFvarLabels *(*selectFeatureData*)*, 159
rmFeaturesOfInterest *(*FeaturesOfInterest-class*)*, 49
rmFeaturesOfInterest,FoICollection,numeric-method *(*FeaturesOfInterest-class*)*, 49
rmFeaturesOfInterest-methods *(*FeaturesOfInterest-class*)*, 49
round, *132*
rownames<-,MChromatograms-method *(*MChromatograms*)*, 67
rtime *(*Spectrum-class*)*, 162
rtime,Chromatogram-method *(*Chromatogram*)*, 14
rtime,MSmap-method *(*MSmap-class*)*, 82
rtime,MSpectra-method *(*MSpectra*)*, 99

rtime,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
rtime,pSet-method (pSet-class), 133
rtime,Spectrum-method (Spectrum-class),
162

sampleNames,MChromatograms-method
(MChromatograms), 67
sampleNames,pSet-method (pSet-class),
133
sampleNames<-,MChromatograms,ANY-method
(MChromatograms), 67
sampleNames<-,pSet,character-method
(pSet-class), 133
samples,MIAPE-method (MIAPE-class), 78
sapply,MSnSetList-method
(MSnSetList-class), 96
scale, *109, 110*
scale,MSnSet-method
(normalise-methods), 109
scanIndex (Spectrum-class), 162
scanIndex,MSpectra-method (MSpectra), 99
scanIndex,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
scanIndex,pSet-method (pSet-class), 133
scanIndex,Spectrum-method
(Spectrum-class), 162
selectFeatureData, 159
setMSnbaseFastLoad (MSnbaseOptions), 84
setMSnbaseParallelThresh
(MSnbaseOptions), 84
setMSnbaseVerbose (MSnbaseOptions), 84
show,Chromatogram-method
(Chromatogram), 14
show,FeatComp-method (FeatComp-class),
47
show,FeaturesOfInterest-method
(FeaturesOfInterest-class), 49
show,FoICollection-method
(FeaturesOfInterest-class), 49
show,MChromatograms-method
(MChromatograms), 67
show,MIAPE-method (MIAPE-class), 78
show,MSmap-method (MSmap-class), 82
show,MSnExp-method (MSnExp-class), 85
show,MSnProcess-method
(MSnProcess-class), 89
show,MSnSet-method (MSnSet-class), 90
show,MSnSetList-method
(MSnSetList-class), 96
show,MSpectra-method (MSpectra), 99
show,MzTab-method (MzTab-class), 105
show,NAnnotatedDataFrame-method
(Deprecated), 39
show,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
show,ProcessingStep-method
(ProcessingStep-class), 132
show,ReporterIons-method
(ReporterIons-class), 157
show,Spectrum-method (Spectrum-class),
162
SimpleList, *102*
smallMolecules (MzTab-class), 105
smooth, *11*, *36*, *87*, *122*, *164*
smooth (smooth-methods), 161
smooth(), *102, 103*
smooth,MSnExp-method (MSnExp-class), 85
smooth,MSpectra-method (MSpectra), 99
smooth,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
smooth,Spectrum-method
(Spectrum-class), 162
smooth-methods, 161
smoothed (Spectrum-class), 162
smoothed,MSpectra-method (MSpectra), 99
smoothed,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114
smoothed,pSet-method (pSet-class), 133
smoothed,Spectrum-method
(Spectrum-class), 162
smoothed<- (Spectrum-class), 162
smoothed<-,OnDiskMSnExp,logical-method
(OnDiskMSnExp-class), 114
smoothed<-,pSet,ANY-method
(pSet-class), 133
smoothed<-,pSet,logical-method
(pSet-class), 133
smoothed<-,Spectrum,ANY-method
(Spectrum-class), 162
smoothed<-,Spectrum,logical-method
(Spectrum-class), 162
spectra (pSet-class), 133
spectra,MSnExp-method (MSnExp-class), 85
spectra,OnDiskMSnExp-method
(OnDiskMSnExp-class), 114

spectra,pSet-method *(pSet-class)*, 133
Spectra::Spectra, *45*
Spectra::Spectra(), *45*
spectrapply *(pSet-class)*, 133
spectrapply(), *84*
spectrapply,OnDiskMSnExp-method *(OnDiskMSnExp-class)*, 114
spectrapply,pSet-method *(pSet-class)*, 133
Spectrum, *30*, *36*, *37*, *45*, *85*, *99*, *101*, *102*, *109*, *123*, *125*, *133*, *140*, *144*, *145*, *157*, *165*, *166*, *169*
Spectrum *(Spectrum-class)*, 162
Spectrum-class, 162
Spectrum1, *37*, *45*, *85*, *86*, *99*, *102*, *117*, *124*, *134*, *162*, *164–166*
Spectrum1 *(Spectrum1-class)*, 165
Spectrum1-class, 165
Spectrum2, *12*, *13*, *37*, *45*, *62*, *79*, *86*, *99*, *102*, *109*, *123*, *124*, *134*, *157*, *162*, *164*, *165*, *167*
Spectrum2 *(Spectrum2-class)*, 165
Spectrum2-class, 165
split,MSnSet,character-method *(MSnSetList-class)*, 96
split,MSnSet,factor-method *(MSnSetList-class)*, 96
splitByFile *(MSnExp-class)*, 85
splitByFile,MSnExp,factor-method *(MSnExp-class)*, 85
splitByFile,OnDiskMSnExp,factor-method *(MSnExp-class)*, 85
strsplit, *55*
supsmu, *42*, *122*

t,MSmap-method *(MSmap-class)*, 82
t.MSnSet *(MSnSet-class)*, 90
tic *(Spectrum-class)*, 162
tic,MSpectra-method *(MSpectra)*, 99
tic,OnDiskMSnExp-method *(OnDiskMSnExp-class)*, 114
tic,pSet-method *(pSet-class)*, 133
tic,Spectrum-method *(Spectrum-class)*, 162
TMT10 *(TMT6)*, 167
TMT10ETD *(TMT6)*, 167
TMT10HCD *(TMT6)*, 167
TMT11 *(TMT6)*, 167
TMT11HCD *(TMT6)*, 167
TMT16 *(TMT6)*, 167
TMT16HCD *(TMT6)*, 167
TMT6, *62*, *159*, 167
TMT6b *(TMT6)*, 167
TMT7 *(TMT6)*, 167
TMT7b *(TMT6)*, 167
topN, *113*
topN *(MSnSet-class)*, 90
topN,matrix-method *(MSnSet-class)*, 90
topN,MSnSet,MSnSet-method *(MSnSet-class)*, 90
topN,MSnSet-method *(MSnSet-class)*, 90
transformIntensity *(Chromatogram)*, 14
transformIntensity,Chromatogram-method *(Chromatogram)*, 14
transformIntensity,MChromatograms-method *(MChromatograms)*, 67
trimMz, *11*, *25*, *36*, *87*, *118*, *122*, *155*, *161*, *164*
trimMz *(trimMz-methods)*, 168
trimMz,MSnExp,numeric-method *(MSnExp-class)*, 85
trimMz,MSnExp-method *(MSnExp-class)*, 85
trimMz,OnDiskMSnExp,numeric-method *(OnDiskMSnExp-class)*, 114
trimMz,Spectrum,numeric-method *(Spectrum-class)*, 162
trimMz,Spectrum-method *(Spectrum-class)*, 162
trimMz-methods, 168
trimws, *93*
trimws *(MSnSet-class)*, 90
trimws,data.frame-method *(MSnSet-class)*, 90
trimws,MSnSet-method *(MSnSet-class)*, 90

unique1 *(FeatComp-class)*, 47
unique1,FeatComp-method *(FeatComp-class)*, 47
unique1,methods *(FeatComp-class)*, 47
unique2 *(FeatComp-class)*, 47
unique2,FeatComp-method *(FeatComp-class)*, 47
unique2,methods *(FeatComp-class)*, 47
unsplit,MSnSetList,factor-method *(MSnSetList-class)*, 96
updateFeatureNames *(MSnSet-class)*, 90
updateFvarLabels *(MSnSet-class)*, 90
updateObject, *169*

updateObject,MSnExp-method
(updateObject-methods), 169
updateObject,Spectrum-method
(updateObject-methods), 169
updateObject-methods, 169
updateSampleNames (MSnSet-class), 90

validateOnDiskMSnExp
(OnDiskMSnExp-class), 114
varLabels,pSet-method (pSet-class), 133
varMetadata,pSet-method (pSet-class),
133
Versioned, *50*, *80*, *86*, *90, 91*, *115*, *133, 134*,
158, *163*, *165, 166*
VersionedBiobase, *86*, *91*, *115*, *134*
Versions, *86*, *91*, *115*, *134*
vsn2, *109*

whichNA (makeNaData), 65
width (ReporterIons-class), 157
width,ReporterIons-method
(ReporterIons-class), 157
width-method (ReporterIons-class), 157
write, *89*
write.exprs, *95*
write.exprs (MSnSet-class), 90
write.exprs,MSnSet-method
(MSnSet-class), 90
writeMgfData, *144*
writeMgfData (writeMgfData-methods), 169
writeMgfData,MSnExp-method
(writeMgfData-methods), 169
writeMgfData,MSpectra-method
(MSpectra), 99
writeMgfData,Spectrum-method
(writeMgfData-methods), 169
writeMgfData-methods, 169
writeMSData
(writeMSData,MSnExp,character-method),
170
writeMSData(), *33*
writeMSData,MSnExp,character-method,
170
writeMzTabData, *151*, 172