Package 'MSnbase'

September 24, 2012

Title MSnbase: Base Functions and Classes for MS-based Proteomics

Version 1.4.1

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

Author@R c(person(given = ''Laurent'', family = ''Gatto'', email = ''lg390@cam.ac.uk'', role = c(''aut'',''cre'')), person(given = ''Guangchuang'', family ='Yu'', email = ''guangchuangyu@gmail.com'',role = ''ctb''))

Author Laurent Gatto <lg390@cam.ac.uk> with contributions from Guangchuang Yu <guangchuangyu@gmail.com>

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

Depends R (>= 2.10), methods, BiocGenerics (>= 0.1.3), Biobase (>= 2.15.2), ggplot2, mzR

Imports graphics, plyr, IRanges, preprocessCore, vsn, grid, reshape

Suggests testthat, zoo, pgfSweave, Rdisop

Enhances foreach, doMC

License Artistic-2.0

LazyLoad yes

LazyData yes

biocViews Infrastructure, Bioinformatics, Proteomics, MassSpectrometry

R topics documented:

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

Description

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

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

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

Author(s)

Laurent Gatto

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

References

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

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

See Also

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

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

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

---

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

Description

This method cleans out individual spectra (`Spectrum` instances) or whole experiments (`MSnExp` instances) of 0-intensity peaks. Orignal 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", 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")` Cleans the `Spectrum` object. Returns a cleaned `Spectrum` instance.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

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

Examples

```
int <- c(1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0)
sp1 <- new("Spectrum2",
           intensity=int,
           mz=1:length(int))
sp2 <- clean(sp1)
intensity(sp1)
intensity(sp2)

file <- dir(system.file(package="MSnbase",dir="extdata"),
            full.name=TRUE,pattern="mzXML$")
aa <- readMSData(file,verbose=FALSE)
bb <- clean(aa,verbose=FALSE)
sum(peaksCount(aa))
sum(peaksCount(bb))
processingData(bb)
```

`combineFeatures` *Combines features in an 'MSnSet' object*

Description

This function combines the features in an "MSnSet" instance applying a summarisation function (see `fun` argument) to sets of features as defined by a factor (see `groupBy` argument).

Usage

```
combineFeatures(object, groupBy, fun = c("mean", "median", "weighted.mean", "sum", "medpolish"),
```

Arguments

`object`	An instance of class "MSnSet" whose features will be summerised.
`groupBy`	An object of class factor defining how to summerise the features.
`fun`	The summerising function. Currently, mean, median, weighted mean, sum and median polish are implemented, but user-defined functions can also be supplied.
`...`	Additional arguments for the `fun` function.
`verbose`	A `logical` indicating whether verbose output is to be printed out.

Value

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

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

```
fn <- dir(system.file(package = "MSnbase", dir = "extdata"),
          full.names = TRUE,
          pattern = "mzXML$")
aa <- readMSData(fn)
aa.quant <- quantify(aa, method = "max", reporters = iTRAQ4)
dim(aa.quant)
## arbitrary grouping into two groups
grp <- as.factor(c(1,1,2,2,2))
aa.quant.comb <- combineFeatures(aa.quant, grp, "sum")
dim(aa.quant.comb)
exprs(aa.quant.comb)
```

`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 <lg390@cam.ac.uk>

See Also

`extractSpectra`

Examples

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

extractSpectra-methods

Extracts a set of spectra from an 'MSnExp' object

Description

Extracts the spectra defined by the `selected` parameter and returns a subset "MSnExp" instance. The `featureData` slot is also subset accordingly.

Note that values in `selected` are recycled if shorter than `length(MSnExp)`.

This method is equivalent as using the `[` operator.

Methods

`signature(object = "MSnExp", selected = "logical")` Returns an "MSnExp" object containing spectra defined by the logical `selected` argument.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

`extractPrecSpectra`

Examples

```
fn <- dir(system.file(package = "MSnbase", dir = "extdata"),
          full.names = TRUE,
          pattern = "mzXML$")
aa <- readMSData(fn, verbose=FALSE)
sel <- rep(FALSE, length(aa))
sel[1:3] <- TRUE
bb <- extractSpectra(aa, sel)
stopifnot(length(bb) == sum(sel))
processingData(bb)
## equivalent
cc <- aa[sel]
all(featureNames(cc) == featureNames(bb))
```

fillUp

Fills up a vector

Description

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

Usage

```
fillUp(x)
```

Arguments

x	a vector.

Value

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

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

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

---

formatRt *Format Retention Time*

Description

Converts seconds to min:sec format

Usage

```
formatRt(rt)
```

Arguments

rt	retention in in seconds

Details

This function is used to convert retention times, expressed in seconds, in the more human friendly format mm:sec.

Value

A character string in mm:ss format

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

```
formatRt(1524)
```

---

iTRAQ4 *iTRAQ 4-plex set*

Description

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

Usage

```
iTRAQ4
iTRAQ5
iTRAQ8
iTRAQ9
```

References

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

See Also

TMT6.

Examples

```
iTRAQ4
iTRAQ4[1:2]

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

itraqdata *Example* MSnExp *data set*

Description

This 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.

itraqdata 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.

This example data set is used in the MSnbase-demo vignette, available with vignette("MSnbase-demo",package="MSn

Usage

```
itraqdata
```

Examples

```
itraqdata
```

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 does not applies.

dateStamp: Object of class character, giving yhe 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.

`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 analyser type (Quadrupole, time-of-flight, ion trap, ...).

`analyserDetails`: Object of class `character` describing the relevant details about the analyser. See MIAPE-MI docuement 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`.

`samples(MIAPE)`: An accessor function for `samples`.

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

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

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

Extends

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

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

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

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

Description

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

This class extends the virtual "pSet" class.

Objects from the Class

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

Slots

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

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

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

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

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

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

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

Extends

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

Methods

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

clean `signature(object = "MSnExp")`: Removes unused 0 intensity data points. See clean 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.

extractSpectra `signature(object = "MSnExp", selected = "logical")`: extracts spectra corresponding to 'TRUE' in selected. See extractSpectra documentation for more details and examples.

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

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

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

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

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

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

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

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

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

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

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

See Also

`"pSet"` and `readMSData` for loading `mzXML`, `mzData` or `mzML` files to generate an instance of `MSnExp`.

Examples

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

MSnProcess-class *The "MSnProcess" Class*

Description

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

Slots

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

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

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

`cleaned`: Object of class `"logical"` indicating whether spectra have been cleaned. See `clean` for more details and examples.

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

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

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

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

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

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

Extends

Class `"Versioned"`, directly.

Methods

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

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

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

Note

This class is likely to be updated using an `AnnotatedDataFrame`.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

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

Examples

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

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

Description

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

Objects from the Class

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

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

Slots

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

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

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

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

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

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

`annotation`: not used here.

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

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

Extends

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

Methods

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

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

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

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

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

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

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

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

meanSdPlot `signature(object = "MSnSet")`: Plots row standard deviations versus row means. See `meanSdPlot` (`vsn` package) 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 with a warning.

as(,"ExpressionSet") `signature( x = "MSnSet")`: Coerce object from `MSnSet` to `ExpressionSet-class`. The `experimentData` slot is dropped.

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

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

topN signature(object = "MSnSet", groupBy, n = 3, fun, ...) Selects the `n` most intense features (typically peptides or spectra) out of all available for each set defined by `groupBy` (typically proteins) and creates a new instance of class `MSnSet`. If less than `n` features are available, all are selected. The `ncol(object)` features are 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. Note that the `qual` slot and the processing information are silently dropped. (Works also with `matrix` instances.)

filterNA signature(object = "MSnSet", pNA = "numeric") This methods subsets `object` by removing features that have more than `pNA` percent of NA values. Default `pNA` is 1/2. To remove any feature that exhibits missing data, use `pNA = 0`. This method also accepts `matrix` instances.

See also the `is.na.MSnSet` and `plotNA` methods for missing data exploration.

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 `..`

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

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

Examples

```
mzxmlfile <- dir(system.file("extdata",package="MSnbase"),
                 pattern="mzXML",full.names=TRUE)
msnexp <- readMSData(mzxmlfile,verbose=FALSE)
msnset <- quantify(msnexp,method="trap",reporters=iTRAQ4,verbose=FALSE)
msnset
```

NAnnotatedDataFrame-class

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

Description

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

Objects from the Class

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

Slots

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

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

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

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

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

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

Extends

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

Methods

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

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

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

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

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

"AnnotatedDataFrame".

Examples

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

---

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

Description

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

Raw spectra and experiments are normalised using max or sum only. Each peak intensity is divided by the highest intensity in the spectrum or the sum of intensities. These methods aim at facilitating relative peaks heights between different spectra.

The method parameter for "MSnSet" can be one of sum, max, quantiles, quantiles.robust or vsn. For sum and max, each feature's reporter intensity is divided by the maximum of the sum respectively. Using quantiles or quantiles.robust uses (robust) quantile normalisation, as implemented in normalize.quantiles and normalize.quantiles.robust of the preprocessCore package. vsn uses the vsn2 from the vsn package. Note that the latter also glog-transforms the intensities. See respective manuals for more details and function arguments.

Arguments

object	An object of class "Spectrum", "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

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.

Examples

```
## quantifying full experiment
data(itraqdata)
qnt <- quantify(itraqdata,method="trap",reporters=iTRAQ4)
qnt.nrm <- normalise(qnt,"quantiles")
qnt.nrm
```

---

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

Description

These method plot mass spectra MZ values against the intensities as line plots. 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 `extractSpectra` or `extractPrecSpectra` methods.

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

Arguments

`x`	Objects of class "Spectrum" or "MSnExp" to be plotted.
`y`	Not used in these methods.
`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 sticks, 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 sticks for centroided reporter ions plots. Default is 0.01.

Methods

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

`signature(x = "Spectrum", y = "missing", reporters = "ReporterIons", full = "logical", centroide`
: 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.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

```
file <- dir(system.file(package="MSnbase",dir="extdata"),full.name=TRUE,pattern="mzXML$")
aa <- readMSData(file,verbose=FALSE)
## plotting experiments
plot(aa[1:2],reporters=iTRAQ4)
plot(aa[1:2],full=TRUE)
## plotting spectra
plot(aa[[1]],reporters=iTRAQ4,full=TRUE)
```

---

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, "tic" (total ion count), "file" (raw data file), "peaks.count" (peaks count) or "charge" (precursor charge).
alpha	Numeric [0,1] indicating transparence level of points.
plot	A logical indicating whether the plot should be printed (default is 'TRUE').

Methods

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

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

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

The plotDensity and plotMzDelta methods for other QC plots.

Examples

```
itraqdata
plot2d(itraqdata,z="tic")
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 'tic', 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, "tic" (total ion count), "peaks.count" (peaks count) or "precursor.mz" (precursor MZ).
log	Logical, whether to log transform the data (default is 'FALSE').
plot	A logical indicating whether the plot should be printed (default is 'TRUE').

Methods

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

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

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

The plot2d and plotDensity methods for other QC plots.

Examples

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

plotMzDelta-methods *The delta m/z plot for 'MSnExp' quality assessment*

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"` 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.
`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 `numeric` 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 a (invisibly) returns the m/z delta histogram.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

Foster JM, Degroeve S, Gatto L, Visser M, Wang R, Griss J, Apweiler R, Martens L. "A 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,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 <lg390@cam.ac.uk>

See Also

See also the `filterNA` method to filter out features with a specified proportion if missing values.

Examples

```
xx <- quantify(itraqdata, reporters = iTRAQ4, verbose = FALSE)
exprs(xx)[sample(prod(dim(xx)), 120)] <- NA

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

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

---

precSelection *Number of precursor selection events*

Description

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

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

Usage

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

Arguments

`object`	An instane of class "`MSnExp`".
`n`	The number of decimal places to round the precursor MZ to. Is passed to the round function.

Value

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

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

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

---

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

Description

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

Objects from the Class

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

Slots

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

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

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

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

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

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

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

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

Extends

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

Methods

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

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

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

abstract Access abstract in `experimentData`.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

spectra `signature(x = "pSet")`: Access the `assayData` slot, returning the features as a `list`.

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.

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

intensity `signature(object = "pSet")`: Accessor for spectra instenities, returned as named list.

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.

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.

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

rtime `signature(object = "pSet")`: Accessor for spectra retention times.

tic `signature(object = "pSet")`: Accessor for spectra total ion counts.

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).

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

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

See Also

`"MSnExp"` for an instantiatable application of `pSet`.

Examples

```
showClass("pSet")
```

---

purityCorrect-methods *Performs reporter ions purity correction*

Description

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

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

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

Function `makeImpuritiesMatrix` creates a simple graphical interface to guide the user in the creation of such a matrix. The function takes the dimension of the square matrix (i.e the number of reporter ions) as argument. When available, default values taken from manufacturer's certificaiton sheets are provided, but batch specific values should be used whenever possible. `makeImpuritiesMatrix` returns the (possibly updated) matrix to be used with `purityCorrect`.

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

0.920	0.020	0.000	0.000
0.059	0.923	0.030	0.001
0.002	0.056	0.924	0.040
0.000	0.001	0.045	0.923

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

Methods

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

Examples

```
## quantifying full experiment
file <- dir(system.file(package="MSnbase",dir="extdata"),full.name=TRUE,pattern="mzXML$")
aa <- readMSData(file,verbose=FALSE)
msnset <- quantify(aa,method="trap",reporters=iTRAQ4)
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)
## or, using makeImpuritiesMatrix()
## Not run: impurities <- makeImpuritiesMatrix(4)
msnset.crct <- purityCorrect(msnset,impurities)
head(exprs(msnset))
head(exprs(msnset.crct))
processingData(msnset.crct)
```

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

Description

This method quantifies individual "Spectrum" objects or full "MSnExp" experiments. Current implementation quantify specific peaks in individual spectra. The peaks of interest are defined by the reporters parameter using one of the possible method methods. This essentially qualifies for MSMS quantitation using isobaric tags, altough any peak can be defined in reporters.

Arguments

object	Objects of class "Spectrum" or "MSnExp".
method	Peak quantitation method. 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).
reporters	An object of class "ReporterIons" that defines the peak(s) to be quantified.
strict	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.
verbose	Verbose of the output (only for MSnExp objects).

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). This is illustrated in the example below.

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.

Methods

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

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.

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

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

Quantifies peaks defined in reporters using method in the Spectrum object.

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

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

upper MZ values for the curve ('lowerMz' and 'upperMz'), reporter ('reporter') and precursor MZ value ('precursor') when available.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

Examples

```
## quantifying full experiment
file <- dir(system.file(package="MSnbase",dir="extdata"),full.name=TRUE,pattern="mzXML$")
aa <- readMSData(file,verbose=FALSE)
msnset <- quantify(aa,method="trap",reporters=iTRAQ4)
msnset
## checking for non-quantified peaks
sum(is.na(exprs(msnset)))
## checking for peaks outside of first reporter width
lowerMz.114 <- qual(msnset)$lowerMz[qual(msnset)$reporter=="114.1"]
upperMz.114 <- qual(msnset)$upperMz[qual(msnset)$reporter=="114.1"]
any(lowerMz.114 < mz(iTRAQ4[1]) - width(iTRAQ4[1]))
any(upperMz.114 > mz(iTRAQ4[1]) + width(iTRAQ4[1]))
## quantifying single spectrum
qty <- quantify(aa[[1]],method="trap",iTRAQ4[1])
qty$peakQuant
qty$curveStats
```

---

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

Description

Reads an ispy2 tab-delimited spreadsheet and generates the corresponding `MSnSet` object.

Usage

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

Arguments

`file`	A `character`, indicating the file name to be read in. Default is "ispy_results.tsv".
`uniquePeps`	A `logical`, indicating whether only unique peptides should be included. Default is TRUE.
`pep`	A `numeric` indicating the posterior error probability threshold for peptides to be considered correctly identified. Default is 0.05.
`na.rm`	A `logical` indicating whether reporter ions containing one or more NA values should be excluded. Default is TRUE.
`min.int`	A `numeric` indicating the minimal summed intensity threshold for reporter data to be imported. Default is 0. Note that 'NA' values are excluded when summing the values.

reporters	A numeric indicating column indices of reporter ions quantitation data. Default is 19:23 for iTRAQ 4-plex.
keepAll	A logical that defines whether all features of the ispy result should be imported. If 'TRUE', 'pep', 'na.rm' and 'min.int' are ignored. This is equivalent to 'pep=1', 'na.rm=FALSE' and 'min.int=0'. Default is 'FALSE'.
verbose	A logical indicating whether verbose output is to be printed out.

Value

An object of class "MSnSet".

Author(s)

Laurent Gatto

References

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

See Also

readMSData to import raw data.

Examples

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

---

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

Description

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

Usage

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

Arguments

file	character vector with file name to be read.
pdata	an object of class "NAnnotatedDataFrame".
smoothed	Logical indicating whether spectra already smoothed or not. Default is 'FALSE'. Used to initialise "MSnProcess" object in processingData slot.
centroided	Logical indicating whether spectra are centroided or not. Default is 'TRUE'. Used to initialise "MSnProcess" object in processingData slot.
cache	Numeric indicating caching level. Default is 1. Under development.
verbose	verbosity flag.

Details

Note that when reading an mgf file, the original order of the spectra is lost. Thus, if the data was originally written to mgf from an MSnExp object using writeMgfData, although the feature names will be identical, the spectra are not as a result of the reordering. See example below.

Value

An instance of

Author(s)

Guangchuang Yu <guangchuangyu@gmail.com> and Laurent Gatto <lg390@cam.ac.uk>

See Also

writeMgfData method to write the content of "Spectrum" or "MSnExp" objects to mgf files. Raw data files can also be read with the readMSData function.

Examples

```
## Not run:
  data(itraqdata)
  writeMgfData(itraqdata,file="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

## End(Not run)
```

---

readMSData *Imports mass-spectrometry raw data files as 'MSnExp' instances.*

Description

Reads as set of XML-based mass-spectrometry data files and generates an "MSnExp" object. This function uses the functionality provided by the mzR package to access data and meta data in mzData, mzXML and mzML.

Usage

```
readMSData(files, pdata = NULL, msLevel = 2, verbose = TRUE,
centroided = FALSE, smoothed = FALSE, removePeaks = 0, clean = FALSE,
cache = 1)
```

Arguments

files	character vector with file names to be read.
pdata	an object of class "NAnnotatedDataFrame".
msLevel	MS level spectra to be read. Use '1' for MS1 spectra of any larger numeric for MSn spectra. Default is '2'.
centroided	Logical indicating whether spectra are centroided or not. Default is 'FALSE'. Used to initialise "MSnProcess" object in processingData slot.
smoothed	Logical indicating whether spectra already smoothed or not. Default is 'FALSE'. Used to initialise "MSnProcess" object in processingData slot.
removePeaks	If > 0 (default), all peaks less or equal then value will set to 0. See removePeaks for more details and examples.
clean	Logical indicating whether 0 intensity peaks should be discarded from spectra. Useful is removePeaks is set. Default is 'FALSE'. See clean for more details and examples.
cache	Numeric indicating caching level. Default is 0. Under development.
verbose	verbosity flag.

Value

An "MSnExp" object.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

"MSnExp" or readMgfData to read mgf peak lists.

Examples

```
file <- dir(system.file(package="MSnbase",dir="extdata"),
            full.name=TRUE,
            pattern="mzXML$")
aa <- readMSData(file)
aa
```

---

readMSnSet *Read 'MSnSet'*

Description

This function reads data files to generate an MSnSet instance. It is a wrapper around Biobase's readExpressionSet function with an additional featureDataFile parameter to include feature data. See also readExpressionSet for more details.

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
           featureDataArgs = list(sep = sep, header = header, row.names = row.names, quote = quo
           experimentDataArgs = list(sep = sep, header = header, row.names = row.names, quote =
           sep = "\t",
           header = TRUE,
           quote = "",
           stringsAsFactors = FALSE,
           row.names = 1L,
           widget = getOption("BioC")$Base$use.widgets, ...)
```

Arguments

	Arguments direclty passed to `readExpressionSet`. The description is from the `readExpressionSet` documentation page.
	(character) File or connection from which to read expression values. The file should contain a matrix with rows as features and columns as samples. `read.table` is called with this as its `file` argument and further arguments given by `exprsArgs`.
~~phenoDataFile~~	(character) File or connection from which to read phenotypic data. `read.AnnotatedDataFrame` is called with this as its `file` argument and further arguments given by `phenoDataArgs`.
`experimentDataFile`	(character) File or connection from which to read experiment data. `read.MIAME` is called with this as its `file` argument and further arguments given by `experimentDataArgs`.
`notesFile`	(character) File or connection from which to read notes; `readLines` is used to input the file.
`path`	(optional) directory in which to find all the above files.
`annotation`	(character) A single character string indicating the annotation associated with this ExpressionSet.
`exprsArgs`	A list of arguments to be used with `read.table` when reading in the expression matrix.
`phenoDataArgs`	A list of arguments to be used (with `read.AnnotatedDataFrame`) when reading the phenotypic data.
`experimentDataArgs`	A list of arguments to be used (with `read.MIAME`) when reading the experiment data.
`sep, header, quote, stringsAsFactors, row.names`	arguments used by the `read.table`-like functions.
`widget`	A boolean value indicating whether widgets can be used. Widgets are NOT yet implemented for `read.AnnotatedDataFrame`.
`...`	Further arguments that can be passed on to the `read.table`-like functions.
	Additional argument, specific to `readMSnSet`:

featureDataFile
: (character) File or connection from which to read feature data. read.AnnotatedDataFrame is called with this as its file argument and further arguments given by phenoDataArgs.

featureDataArgs
: A list of arguments to be used (with read.AnnotatedDataFrame) when reading the phenotypic data.

Value

An instance of the MSnSet class.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

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)
```

---

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.

Note that the number of peaks is not changed; the peaks below the threshold are set to 0 and the object is not cleanded out (see clean). An illustrative example is shown below.

Methods

signature(object = "MSnExp", t, verbose = "logical") Removes low intensity peaks of all spectra in MSnExp object. t sets the minimum peak intensity. Default is "min", i.e the smallest intensity in each spectrum. Other numeric values are valid. Displays a control bar if verbose set to TRUE (default). Returns a new MSnExp instance.

signature(object = "Spectrum", t) Removes low intensity peaks of Spectrum object. t sets the minimum peak intensity. Default is "min", i.e the smallest intensity in each spectrum. Other numeric values are valid. Returns a new Spectrum instance.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

clean and trimMz for other spectra processing methods.

Examples

```
int <- c(2,0,0,0,1,5,1,0,0,1,3,1,0,0,1,4,2,1)
sp1 <- new("Spectrum2",
           intensity=int,
           mz=1:length(int))
sp2 <- removePeaks(sp1) ## no peaks are removed here
                        ## as min intensity is 1 and
                        ## no peak has a max int <= 1
sp3 <- removePeaks(sp1,3)
intensity(sp1)
intensity(sp2)
intensity(sp3)

peaksCount(sp1) == peaksCount(sp2)
peaksCount(sp1) < peaksCount(sp3)

file <- dir(system.file(package="MSnbase",dir="extdata"),
            full.name=TRUE,pattern="mzXML$")
aa <- readMSData(file)
aa <- removePeaks(aa)
processingData(aa)
```

---

removeReporters-methods

Removes reporter ion tag peaks

Description

This methods sets all the reporter tag ion peaks from one MS2 spectrum or all the MS2 spectra of an experiment to 0. Reporter data is specified using an "ReporterIons" instance. The peaks are selected around the expected reporter ion m/z value +/- the reporter width. Optionally, the spectrum/spectra can be cleaned to remove successive 0 intensity data points (see the clean function for details).

Note that this method only works for MS2 spectra or experiments that contain MS2 spectra. It will fail for MS1 spectrum.

Methods

signature(object = "MSnExp", reporters = "ReporterIons", clean = "logical", verbose = "logical"
: The reporter ion peaks defined in the reporters instance of all the MS2 spectra of the "MSnExp" instance are set to 0 and, if clean is set to TRUE, cleaned. The default value of reporters is NULL, which leaves the spectra as unchanged. The verbose parameter (default is TRUE) defines whether a progress bar should be showed.

signature(object = "Spectrum", reporters = "ReporterIons", clean = "FALSE") The reporter ion peaks defined in the reporters instance of MS2 "Spectrum" instance are set to 0 and, if clean is set to TRUE, cleaned. The default value of reporters is NULL, which leaves the spectrum as unchanged.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

clean and removePeaks for other spectra processing methods.

Examples

```
sp1 <- itraqdata[[1]]
sp2 <- removeReporters(sp1,reporters=iTRAQ4)
sel <- mz(sp1) > 114 & mz(sp1) < 114.2
mz(sp1)[sel]
intensity(sp1)[sel]
plot(sp1,full=TRUE,reporters=iTRAQ4)
intensity(sp2)[sel]
plot(sp2,full=TRUE,reporters=iTRAQ4)
```

ReporterIons-class *The "ReporterIons" Class*

Description

The ReporterIons class allows to define a set of isobaric reporter ions that are used for quantification in MSMS mode, e.g. iTRAQ (isobaric tag for relative and absolute quantitation) or TMT (tandem mass tags). ReporterIons instances can them be used when quantifying "MSnExp" data of plotting the reporters peaks based on in "Spectrum2" ojects.

Some reporter ions are provided with MSnbase an can be loaded with the data function. These reporter ions data sets are:

iTRAQ4: ReporterIon object for the iTRAQ 4-plex set. Load with data(iTRAQ4).

iTRAQ5: ReporterIon object for the iTRAQ 4-plex set plus the isobaric tag. Load with data(iTRAQ5).

TMT6: ReporterIon object for the TMT 6-plex set. Load with data(TMT6).

TMT7: ReporterIon object for the TMT 6-plex set plus the isobaric tag. Load with data(TMT6).

Objects from the Class

Objects can be created by calls of the form new("ReporterIons", ...).

Slots

name: Object of class "character" to identify the ReporterIons instance.

reporterNames: Object of class "character" naming each individual reporter of the ReporterIons instance. If not provided 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.

`reporterColours(object)` **or reporterColors(object)** Returns the colours used to highlight the reporter ions.

`reporterNames(object)` Returns the name of the individual reporter ions. If not specified or is an incorrect number of names is provided at initialisation, the names are generated automatically by concatenating the instance name and the reporter's MZ values.

`reporterNames(object) <- value` Sets the reporter names to `value`, which must be a character of the same length as the number of reporter ions.

`width(object)` Returns the widths in which the reporter ion peaks are expected.

`names(object)` Returns the name of the `ReporterIons` object.

`description(object)` Returns the description of the `ReporterIons` object.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

References

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

Thompson A, Sch\"afer J, Kuhn K, Kienle S, Schwarz J, Schmidt G, Neumann T, Johnstone R, Mohammed AK, Hamon C. "Tandem mass tags: a novel quantification strategy for comparative analysis of complex protein mixtures by MS/MS." *Anal Chem.* 2003 Apr 15;75(8):1895-904. *Erratum* in: *Anal Chem.* 2006 Jun 15;78(12):4235. Mohammed, A Karim A [added] and *Anal Chem.* 2003 Sep 15;75(18):4942. Johnstone, R [added]. PubMed PMID: 12713048.

See Also

`TMT6` or `iTRAQ4` for readily available examples.

Examples

```
## Code used for the iTRAQ4 set
ri <- new("ReporterIons",
          description="4-plex iTRAQ",
          name="iTRAQ4",
          reporterNames=c("iTRAQ4.114","iTRAQ4.115",
                          "iTRAQ4.116","iTRAQ4.117"),
          mz=c(114.1,115.1,116.1,117.1),
          col=c("red","green","blue","yellow"),
          width=0.05)
ri
reporterNames(ri)
ri[1:2]
```

`Spectrum-class` *The "Spectrum" Class*

Description

Virtual container for spectrum data common to all different types of spectra. A `Spectrum` object can not be directly instanciated. Use `"Spectrum1"` and `"Spectrum2"` instead.

Slots

`msLevel`: Object of class `"integer"` indicating the MS level: 1 for MS1 level `Spectrum1` objects and 2 for MSMSM `Spectrum2` objects. Levels > 2 have not been tested and will be handled as MS2 spectra.

`peaksCount`: Object of class `"integer"` indicating the number of MZ peaks.

`rt`: Object of class `"numeric"` indicating the retention time (in seconds) for the current ions.

`acquisitionNum`: Object of class `"integer"` corresponding to the acquisition number of the current spectrum.

`scanIndex`: Object of class `"integer"` indicating the scan index of the current spectrum.

`mz`: Object of class `"numeric"` of length equal to the peaks count (see `peaksCount` slot) indicating the MZ values that have been measured for the current ion.

`intensity`: Object of class `"numeric"` of same length as `mz` indicating the intensity at which each `mz` datum has been measured.

`centroided`: Object of class `"logical"` indicating if instance is centroided ('TRUE') of uncentroided ('FALSE').

`fromFile`: Object of class `"integer"` referencing the file the spectrum originates. The file names are stored in the `processingData` slot of the `"MSnExp"` or `"MSnSet"` instance that contains the current `"Spectrum"` instance.

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

Extends

Class `"Versioned"`, directly.

Methods

`acquisitionNum(object)` Returns the acquisition number of the spectrum as an integer.

`centroided(object)` Indicates whether spectrum is centroided ('TRUE') or uncentroided ('FALSE').

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

`fromFile(object)` Returns the index of the raw data file from which the current instances originates as an integer.

`intensity(object)` Returns an object of class `"numeric"` containing the intensities of the spectrum.

`msLevel(object)` Returns an MS level of the spectrum as an integer.

`mz(object)` Returns an object of class `"numeric"` containing the MZ value of the spectrum peaks.

`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.

`tic(object)` Returns the total ion count for the spectrum as a numeric.

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

plot `signature(x = "Spectrum", y = "missing")`: Plots intensity against mz. See `plot.Spectrum` 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.

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.

as `signature(object = "Spectrum", "data.frame")`: Coerces the `Spectrum` object to a two-column `data.frame` containing intensities and MZ values.

Note

This is a virtual class and can not be instanciated directly.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

Instaciable sub-classes `"Spectrum1"` and `"Spectrum2"` for MS1 and MS2 spectra.

Spectrum1-class *The "Spectrum1" Class for MS1 Spectra*

Description

Spectrum1 extends the "Spectrum" class and introduces an MS1 specific attribute in addition to the slots in "Spectrum". Spectrum1 instances are not created directly but are contained in the assayData slot of an "MSnExp".

Slots

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

msLevel: Object of class "integer" indicating the MS level: always 1 in this case (inherited from "Spectrum").

peaksCount: Object of class "integer" indicating the number of MZ peaks (inherited from "Spectrum").

rt: Object of class "numeric" indicating the retention time (in seconds) for the current ions (inherited from "Spectrum").

acquisitionNum: Object of class "integer" corresponding to the acquisition number of the current spectrum (inherited from "Spectrum").

scanIndex: Object of class "integer" indicating the scan index of the current spectrum (inherited from "Spectrum").

mz: Object of class "numeric" of length equal to the peaks count (see peaksCount slot) indicating the MZ values that have been measured for the current ion (inherited from "Spectrum").

intensity: Object of class "numeric" of same length as mz indicating the intensity at which each mz datum has been measured "Spectrum").

.__classVersion__: Object of class "Versions" indicating the versions of the Spectrum and Spectrum1 classes of the instance. Intended for developer use and debugging.

Extends

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

Methods

See "Spectrum" for additional accessors and methods to process Spectrum1 objects.

polarity(object) Returns the polarity of the spectrum as an integer.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

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

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

Description

Spectrum2 extends the "Spectrum" class and introduces several MS2 specific attributes in addition to the slots in "Spectrum". Spectrum2 are not created directly but are contained in the assayData slot of an "MSnExp".

Slots

merged: Object of class "numeric" indicating of how many combination the current spectrum is the result of.

precScanNum: Object of class "integer" indicating the precursor MS scan index in the original input file.

precursorMz: Object of class "numeric" providing the precursor ion MZ value.

precursorIntensity: Object of class "numeric" providing the precursor ion intensity.

precursorCharge: Object of class "integer" indicating the precursor ion charge.

collisionEnergy: Object of class "numeric" indicating the collision energy used to fragment the parent ion.

msLevel: Object of class "integer" indicating the MS level: 2 in this case (inherited from "Spectrum").

peaksCount: Object of class "integer" indicating the number of MZ peaks (inherited from "Spectrum").

rt: Object of class "numeric" indicating the retention time (in seconds) for the current ions (inherited from "Spectrum").

acquisitionNum: Object of class "integer" corresponding to the acquisition number of the current spectrum (inherited from "Spectrum").

scanIndex: Object of class "integer" indicating the scan index of the current spectrum (inherited from "Spectrum").

mz: Object of class "numeric" of length equal to the peaks count (see peaksCount slot) indicating the MZ values that have been measured for the current ion (inherited from "Spectrum").

intensity: Object of class "numeric" of same length as mz indicating the intensity at which each mz datum has been measured "Spectrum").

.__classVersion__: Object of class "Versions" indicating the versions of the Spectrum and Spectrum2 classes of the instance. Intended for developer use and debugging.

Extends

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

Methods

See "Spectrum" for additional accessors and methods for Spectrum2 objects.

precursorMz(object) Returns the precursor MZ value as a numeric.

precursorMz(object) Returns the precursor scan number in the original data file as an integer.

precursorIntensity(object) Returns the precursor intensity as a numeric.

`precursorCharge(object)` Returns the precursor intensity as a integer.

`collisionEnergy(object)` Returns the collision energy as an numeric.

`removeReporters(object, ...)` Removes all reporter ion peaks. See `removeReporters` documentation for more details and examples.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

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

`TMT6` *TMT 6-plex set*

Description

This instance of class "`ReporterIons`" corresponds to the TMT 6-plex set, i.e the 126, 127, 128, 129, 130 and 131 isobaric tags. In the TMT7 data set, an unfragmented tag, i.e reporter and attached isobaric tag, is also included at MZ 229. These objects are used to plot the reporter ions of interest in an MSMS spectra (see "`Spectrum2`") as well as for quantification (see `quantify`).

Usage

```
TMT6
TMT7
```

References

Thompson A, Sch\"afer J, Kuhn K, Kienle S, Schwarz J, Schmidt G, Neumann T, Johnstone R, Mohammed AK, Hamon C. "Tandem mass tags: a novel quantification strategy for comparative analysis of complex protein mixtures by MS/MS." *Anal Chem.* 2003 Apr 15;75(8):1895-904. *Erratum* in: *Anal Chem.* 2006 Jun 15;78(12):4235. Mohammed, A Karim A [added] and *Anal Chem.* 2003 Sep 15;75(18):4942. Johnstone, R [added]. PubMed PMID: 12713048.

See Also

`iTRAQ4`.

Examples

```
TMT6
TMT6[1:2]

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

trimMz-methods *Trims 'MSnExp' or 'Spectrum' instances*

Description

This method selects a range of MZ values in a single spectrum (`Spectrum` instances) or all the spectra of an experiment (`MSnExp` instances). The regions to trim are defined by the range of `mzlim` argument, such that MZ values < `min(mzlim)` and MZ values > `max(mzlim)` are trimmed away.

Methods

`signature(object = "MSnExp", mzlim = "numeric")` Trims all spectra in `MSnExp` object according to `mzlim`. Returns a cleaned `MSnExp` instance.

`signature(object = "Spectrum", mzlim = "numeric")` Trims the `Spectrum` object and retruns a new trimmed object.

Author(s)

Laurent Gatto <lg390@cam.ac.uk>

See Also

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

Examples

```
mz <- 1:100
sp1 <- new("Spectrum2",
           mz=mz,
           intensity=abs(rnorm(length(mz))))
sp2 <- trimMz(sp1,c(25,75))
range(mz(sp1))
range(mz(sp2))
file <- dir(system.file(package="MSnbase",dir="extdata"),
            full.name=TRUE,pattern="mzXML$")
aa <- readMSData(file,verbose=FALSE)
bb <- trimMz(aa,c(113,117))
range(mz(bb))
range(mz(bb))
processingData(bb)
```

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

```
## Not run:
  data(itraqdata)
  writeMgfData(itraqdata,file="itraqdata.mgf",COM="MSnbase itraqdata")
  itraqdata2 <- readMgfData("itraqdata.mgf")
  ## note that the order of the spectra
  ## and precision of some values (precursorMz for instance)
  ## are altered
  match(signif(precursorMz(itraqdata2),4),signif(precursorMz(itraqdata),4))
  ## [1]  1 10 11 12 13 14 15 16 17 18 ...
  ## ... but all the precursors are there
  all.equal(sort(precursorMz(itraqdata2)),sort(precursorMz(itraqdata)),
            check.attributes=FALSE,
            tolerance=10e-5)
  ## is TRUE
  all.equal(as.data.frame(itraqdata2[[1]]),as.data.frame(itraqdata[[1]]))
  ## is TRUE
  all.equal(as.data.frame(itraqdata2[[3]]),as.data.frame(itraqdata[[11]]))
  ## is TRUE
  ## But, beware that
  all(featureNames(itraqdata2)==featureNames(itraqdata))
  ## is TRUE too!

## End(Not run)
```

Index