Package ‘xcms’

November 21, 2016

Version 1.50.0
Date 2016-09-26
Title LC/MS and GC/MS Data Analysis
Author Colin A. Smith <csmith@scripps.edu>,
Ralf Tautenhahn <rtautenh@gmail.com>,
Steffen Neumann <sneumann@ipb-halle.de>,
Paul Benton <hpbenton@scripps.edu>,
Christopher Conley <cjconley@ucdavis.edu>,
Johannes Rainer <Johannes.Rainer@eurac.edu>
Maintainer Steffen Neumann <sneumann@ipb-halle.de>
Depends R (>= 2.14.0), methods, mzR (>= 1.1.6), BiocGenerics,
ProtGenerics, Biobase
Imports lattice, RColorBrewer, plyr, RANN, multtest, MassSpecWavelet
(> = 1.5.2), BiocParallel, S4Vectors
Suggests BiocStyle, knitr (>= 1.1.0), faahKO, msdata, ncdf4, rgl,
RUnit
Enhances Rgraphviz, Rmpi, XML
Description Framework for processing and visualization of chromatographically
separated and single-spectra mass spectral data. Imports from AIA/ANDI NetCDF,
mzXML, mzData and mzML files. Preprocesses data for high-throughput, untargeted
analyte profiling.
License GPL (>= 2) + file LICENSE
URL http://metlin.scripps.edu/download/ and
https://github.com/sneumann/xcms
VignetteBuilder knitr
BugReports https://github.com/sneumann/xcms/issues/new
biocViews MassSpectrometry, Metabolomics
RoxygenNote 5.0.1
Collate 'AllGenerics.R' 'DataClasses.R' 'Deprecated.R' 'MPI.R' 'c.R'
'cwTools.R' 'databases.R' 'functions-binning.R'
'do_detectFeatures-functions.R' 'fastMatch.R'
'functions-utils.R' 'functions-xcmsEIC.R'
'functions-xcmsFragments.R' 'functions-xcmsRaw.R'
'functions-xcmsSet.R' 'init.R' 'matchpeaks.R' 'methods-IO.R'

1
`methods-netCdfSource.R` `methods-rampSource.R`
`methods-xcmsEIC.R` `methods-xcmsFragments.R`
`models.R` `msn2xcmsRaw.R` `mzClust.R` `netCDF.R` `plotQC.R`
`writemztab.R` `xcmsSource.R` `zzz.R`

**R topics documented:**

```
absent-methods .................................................. 4
AutoLockMass-methods ......................................... 4
c-methods .......................................................... 5
calibrate-methods ............................................. 6
collect-methods ................................................. 7
diffreport-methods ........................................... 8
etg ................................................................. 10
fillPeaks-methods ............................................. 11
fillPeaks.chrom-methods ..................................... 11
fillPeaks.MSW-methods ....................................... 12
findMZ ............................................................ 13
findneutral .................................................... 14
findPeaks-methods ............................................ 16
findPeaks.addPredictedIsotopeFeatures-methods .......... 17
findPeaks.centWave-methods .................................. 19
findPeaks.centWaveWithPredictedIsotopeROIs-methods .. 21
findPeaks.massifquant-methods .............................. 23
findPeaks.matchedFilter-methods ............................ 26
findPeaks.MS1-methods ...................................... 27
findPeaks.MSW-methods ...................................... 28
getEIC-methods .................................................. 29
getPeaks-methods ............................................. 30
getScan-methods ................................................ 31
getSpec-methods ................................................ 32
getXcmsRaw-methods .......................................... 32
group-methods ................................................... 33
group.density .................................................... 34
group.mzClust ................................................... 34
group.nearest .................................................... 35
grouppnames-methods ......................................... 37
groupval-methods ............................................... 37
image-methods ................................................... 38
levelplot-methods ............................................. 39
loadRaw-methods ............................................... 39
medianFilter ..................................................... 40
msn2xcmsRaw ...................................................... 41
peakPlots-methods ............................................ 42
peakTable-methods ............................................. 42
plot.xcmsEIC .................................................... 44
plotChrom-methods ............................................ 45
plotEIC-methods ............................................... 45
```
absent-methods  Determine which peaks are absent / present in a sample class

Description
Determine which peaks are absent / present in a sample class

Arguments

  object       xcmsSet-class object
  class        Name of a sample class from sampclass
  minfrac      minimum fraction of samples necessary in the class to be absent/present

Details
Determine which peaks are absent / present in a sample class. The functions treat peaks that are only present because of fillPeaks correctly, i.e. does not count them as present.

Value
An logical vector with the same length as nrow(groups(object)).

Methods

  object = "xcmsSet"  absent(object, ...)  present(object, ...)

See Also
  group  diffreport

AutoLockMass-methods  Automatic parameter for Lock mass fixing AutoLockMass ~

Description
AutoLockMass - This function decides where the lock mass scans are in the xcmsRaw object. This is done by using the scan time differences.

Arguments

  object       An xcmsRaw-class object

Value
AutoLockMass A numeric vector of scan locations corresponding to lock Mass scans

Methods

  object = "xcmsRaw"  signature(object = "xcmsRaw")
### Examples

```r
## Not run: library(xcms)
library(faahKO) ## These files do not have this problem to correct for but just for an example
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
xr<-xcmsRaw(cdffiles[1])
xr

## Lets assume that the lockmass starts at 1 and is every 100 scans
lockMass<-xcms:::makeacqNum(xr, freq=100, start=1)
## these are equalvent
lockmass2<AutoLockMass(xr)
all((lockmass == lockmass2) == TRUE)

ob<-stitch(xr, lockMass)

## End(Not run)
```

---

**c-methods**  
*Combine xcmsSet objects*

---

**Description**

Combines the samples and peaks from multiple `xcmsSet` objects into a single object. Group and retention time correction data are discarded. The `profinfo` list is set to be equal to the first object.

**Arguments**

- `xs1`: `xcmsSet` object
- `...`: `xcmsSet` objects

**Value**

A `xcmsSet` object.

**Methods**

- `xs1 = "xcmsRaw"`  
  
**Author(s)**

Colin A. Smith, <csmith@scripps.edu>

**See Also**

* `xcmsSet-class`
calibrate-methods

Calibrate peaks for correcting unprecise m/z values

Description

Calibrate peaks of a xcmsSet via a set of known masses

Arguments

object a xcmsSet object with uncalibrated mz
calibrants a vector or a list of vectors with reference m/z-values
method the used calibrating-method, see below
mzppm the relative error used for matching peaks in ppm (parts per million)
mzabs the absolute error used for matching peaks in Da
neighbours the number of neighbours from which the one with the highest intensity is used (instead of the nearest)
plotres can be set to TRUE if wanted a result-plot showing the found m/z with the distances and the regression

Value

object a xcmsSet with one or more samples
calibrants for each sample different calibrants can be used, if a list of m/z-vectors is given. The length of the list must be the same as the number of samples, alternatively a single vector of masses can be given which is used for all samples.
method "shift" for shifting each m/z, "linear" does a linear regression and adds a linear term to each m/z. "edgeshift" does a linear regression within the range of the m/z-calibrants and a shift outside.

Methods

object = "xcmsSet" calibrate(object, calibrants, method="linear", mzabs=0.0001, mzppm=5, neighbours=3, plotres=FALSE)

See Also

xcmsSet-class.
collect-methods

Collect MS^n peaks into xcmsFragments

Description

Collecting Peaks into \texttt{xcmsFragments} from several MS-runs using \texttt{xcmsSet} and \texttt{xcmsRaw}.

Arguments

- \texttt{object} \hspace{1cm} (empty) \texttt{xcmsFragments-class} object
- \texttt{x} \hspace{1cm} A \texttt{xcmsSet-class} object which contains picked ms1-peaks from several experiments
- \texttt{compMethod} \hspace{1cm} \{"floor", "round", "none"\}: compare-method which is used to find the parent peak of a MSnpeak through comparing the MZ-values of the MS1peaks with the MSnParentPeaks.
- \texttt{snthresh}, \texttt{mzgap}, \texttt{uniq}
  
  these are the parameters for the getspec-peakpicker included in xcmsRaw.

Details

After running \texttt{collect(xFragments,xSet)} The peak table of the xcmsFragments includes the ms1Peaks from all experiments stored in a xcmsSet-object. Further it contains the relevant msN-peaks from the xcmsRaw-objects, which were created temporarily with the paths in xcmsSet.

Value

A matrix with columns:

- \texttt{peakID} \hspace{1cm} unique identifier of every peak
- MSnParentPeakID \hspace{1cm} PeakID of the parent peak of a msLevel>1 - peak, it is 0 if the peak is msLevel 1.
- msLevel \hspace{1cm} The msLevel of the peak.
- rt \hspace{1cm} retention time of the peak midpoint
- mz \hspace{1cm} the mz-Value of the peak
- intensity \hspace{1cm} the intensity of the peak
- sample \hspace{1cm} the number of the sample from the xcmsSet
- GroupPeakMSn \hspace{1cm} Used for grouped xcmsSet groups
- CollisionEnergy \hspace{1cm} The collision energy of the fragment

Methods

\texttt{object} = \texttt{"xcmsFragments"} \hspace{1cm} \texttt{collect(object, \ldots)}
Create a report showing the most significant differences between two sets of samples. Optionally create extracted ion chromatograms for the most significant differences.

Arguments

- **object**: the xcmsSet object
- **class1**: character vector with the first set of sample classes to be compared
- **class2**: character vector with the second set of sample classes to be compared
- **filebase**: base file name to save report. .tsv file and .eic will be appended to this name for the tabular report and EIC directory, respectively. If blank nothing will be saved
- **eicmax**: number of the most significantly different analytes to create EICs for
- **eicwidth**: width (in seconds) of EICs produced
- **sortpval**: logical indicating whether the reports should be sorted by p-value
- **classeic**: character vector with the sample classes to include in the EICs
- **value**: intensity values to be used for the diffreport. If value="into", integrated peak intensities are used. If value="maxo", maximum peak intensities are used. If value="intb", baseline corrected integrated peak intensities are used (only available if peak detection was done by findPeaks.centWave).
- **metlin**: mass uncertainty to use for generating link to Metlin metabolite database. The sign of the uncertainty indicates negative or positive mode data for M+H or M-H calculation. A value of FALSE or 0 removes the column
- **h**: Numeric variable for the height of the eic and boxplots that are printed out.
- **w**: Numeric variable for the width of the eic and boxplots print out made.
- **mzdec**: Number of decimal places of title m/z values in the eic plot.
- **...**: optional arguments to be passed to mt.teststat

Details

This method handles creation of summary reports with statistics about which analytes were most significantly different between two sets of samples. It computes Welch’s two-sample t-statistic for each analyte and ranks them by p-value. It returns a summary report that can optionally be written out to a tab-separated file.

Additionally, it does all the heavy lifting involved in creating superimposed extracted ion chromatograms for a given number of analytes. It does so by reading the raw data files associated with the samples of interest one at a time. As it does so, it prints the name of the sample it is currently reading. Depending on the number and size of the samples, this process can take a long time.

If a base file name is provided, the report (see Value section) will be saved to a tab separated file. If EICs are generated, they will be saved as 640x480 PNG files in a newly created subdirectory.
However this parameter can be changed with the commands arguments. The numbered file names correspond to the rows in the report.

Chromatographic traces in the EICs are colored and labeled by their sample class. Sample classes take their color from the current palette. The color a sample class is assigned is dependent its order in the xcmsSet object, not the order given in the class arguments. Thus `levels(sampclass(object))[1]` would use color `palette()[1]` and so on. In that way, sample classes maintain the same color across any number of different generated reports.

When there are multiple sample classes, xcms will produce boxplots of the different classes and will generate a single anova p-value statistic. Like the eic’s the plot number corresponds to the row number in the report.

### Value

A data frame with the following columns:

- **fold**: mean fold change (always greater than 1, see `tstat` for which set of sample classes was higher)
- **tstat**: Welch’s two sample t-statistic, positive for analytes having greater intensity in class2, negative for analytes having greater intensity in class1
- **pvalue**: p-value of t-statistic
- **anova**: p-value of the anova statistic if there are multiple classes
- **mzmed**: median m/z of peaks in the group
- **mzmin**: minimum m/z of peaks in the group
- **mzmax**: maximum m/z of peaks in the group
- **rtmed**: median retention time of peaks in the group
- **rtmin**: minimum retention time of peaks in the group
- **rtmax**: maximum retention time of peaks in the group
- **npeaks**: number of peaks assigned to the group
- **Sample Classes**: number samples from each sample class represented in the group
- **metlin**: A URL to metlin for that mass
- **Sample Names**: integrated intensity value for every sample

### Methods

```r
object = "xcmsSet" diffreport(object, class1 = levels(sampclass(object))[1], class2 = levels(sampclass(object))[2], filebase = ..., sortpval = TRUE, classeic = c(class1,class2), value=c("into","maxo","intb"), metlin = FALSE, h=480,w=640, mzdec=2, ...)
```

### See Also

- xcmsSet-class
- mt.teststat
- palette
Empirically Transformed Gaussian function

Description

A general function for asymmetric chromatographic peaks.

Usage

etg(x, H, t1, tt, k1, kt, lambda1, lambdat, alpha, beta)

Arguments

- x: times to evaluate function at
- H: peak height
- t1: time of leading edge inflection point
- tt: time of trailing edge inflection point
- k1: leading edge parameter
- kt: trailing edge parameter
- lambda1: leading edge parameter
- lambdat: trailing edge parameter
- alpha: leading edge parameter
- beta: trailing edge parameter

Value

The function evaluated at times x.

Author(s)

Colin A. Smith, <csmith@scripps.edu>

References

Description

For each sample, identify peak groups where that sample is not represented. For each of those peak groups, integrate the signal in the region of that peak group and create a new peak.

Arguments

- **object**: the `xcmsSet` object
- **method**: the filling method

Details

After peak grouping, there will always be peak groups that do not include peaks from every sample. This method produces intensity values for those missing samples by integrating raw data in peak group region. According to the type of raw-data there are 2 different methods available. for filling gcms/lcms data the method "chrom" integrates raw-data in the chromatographic domain, whereas "MSW" is used for peaklists without retention-time information like those from direct-infusion spectra.

Value

A `xcmsSet` objects with filled in peak groups.

Methods

```r
object = "xcmsSet" fillPeaks(object, method="")
```

See Also

- `xcmsSet-class, getPeaks`

---

Description

For each sample, identify peak groups where that sample is not represented. For each of those peak groups, integrate the signal in the region of that peak group and create a new peak.
Arguments

object: the `xcmsSet` object

nSlaves: number of slaves/cores to be used for parallel peak filling. MPI is used if installed, otherwise the snow package is employed for multicore support. If none of the two packages is available it uses the parallel package for parallel processing on multiple CPUs of the current machine.

expand.mz: Expansion factor for the m/z range used for integration.

expand.rt: Expansion factor for the retention time range used for integration.

Details

After peak grouping, there will always be peak groups that do not include peaks from every sample. This method produces intensity values for those missing samples by integrating raw data in peak group region. In a given group, the start and ending retention time points for integration are defined by the median start and end points of the other detected peaks. The start and end m/z values are similarly determined. Intensities can be still be zero, which is a rather unusual intensity for a peak. This is the case if e.g. the raw data was thresholded, and the integration area contains no actual raw intensities, or if one sample is miscalibrated, such that the raw data points are (just) outside the integration area.

Importantly, if retention time correction data is available, the alignment information is used to more precisely integrate the proper region of the raw data. If the corrected retention time is beyond the end of the raw data, the value will be not-a-number (NaN).

Value

A `xcmsSet` objects with filled in peak groups (into and maxo).

Methods

object = "xcmsSet" fillPeaks.chrom(object, nSlaves=0, expand.mz=1, expand.rt=1)

See Also

`xcmsSet-class`, `getPeaks`, `fillPeaks`
Details

After peak grouping, there will always be peak groups that do not include peaks from every sample. This method produces intensity values for those missing samples by integrating raw data in peak group region. In a given group, the start and ending m/z values for integration are defined by the median start and end points of the other detected peaks.

Value

A `xcmsSet` objects with filled in peak groups.

Methods

```r
object = "xcmsSet" fillPeaks.MSW(object)
```

See Also

`xcmsSet-class`, `getPeaks`, `fillPeaks`

---

findMZ  

*Find fragment ions in xcmsFragment objects*

Description

This is a method to find a fragment mass with a ppm window in a `xcmsFragment` object.

Usage

```r
findMZ(object, find, ppmE=25, print=TRUE)
```

Arguments

- `object`: xcmsFragment object type
- `find`: The fragment ion to be found
- `ppmE`: the ppm error window for searching
- `print`: If we should print a nice little report

Details

The method simply searches for a given fragment ion in an `xcmsFragment` object type given a certain ppm error window.

Value

A data frame with the following columns:

- `PrecursorMz`: The precursor m/z of the fragment
- `MSnParentPeakID`: An index ID of the location of the precursor peak in the `xcmsFragment` object
- `msLevel`: The level of the found fragment ion
- `rt`: the Retention time of the found ion
findneutral

mz the actual m/z of the found fragment ion
intensity The intensity of the fragment ion
sample Which sample the fragment ion came from
GroupMSn an ID if the peaks were grouped by an xcmsSet grouping
CollisionEnergy The collision energy of the precursor scan

Author(s)
H. Paul Benton, <hpaul.beonton08@imperial.ac.uk>

References

See Also
findneutral.

Examples
## Not run:
library(msdata)
mzdatapath <- system.file("iontrap", package = "msdata")
mzdatafiles<-list.files(mzdatapath, pattern = "extracted.mzData", recursive = TRUE, full.names = TRUE)
xs <- xcmsSet(mzdatafiles, method = "MS1")
## takes only one file from the file set
xfrag <- xcmsFragments(xs)
found<-findMZ(xfrag, 657.3433, 50)
## End(Not run)

---

findneutral  
Find neutral losses in xcmsFragment objects

Description
This is a method to find a neutral loss with a ppm window in a xcmsFragment object

Usage
findneutral(object, find, ppmE=25, print=TRUE)

Arguments

object xcmsFragment object type
find The neutral loss to be found
ppmE the ppm error window for searching
print If we should print a nice little report
findneutral

Details

The method searches for a given neutral loss in an xcmsFragment object type given a certain ppm error window. The neutral losses are generated between neighbouring ions. The resulting data frame shows the whole scan in which the neutral loss was found.

Value

A data frame with the following columns:

- **PrecursorMz**: The precursor m/z of the neutral losses
- **MSnParentPeakID**: An index ID of the location of the precursor peak in the xcmsFragment object
- **msLevel**: The level of the found fragment ion
- **rt**: the Retention time of the found ion
- **mz**: the actual m/z of the found fragment ion
- **intensity**: The intensity of the fragment ion
- **sample**: Which sample the fragment ion came from
- **GroupPeakMSn**: an ID if the peaks were grouped by an xcmsSet grouping
- **CollisionEnergy**: The collision energy of the precursor scan

Author(s)

H. Paul Benton, <hpbenton@scripps.edu>

References


See Also

findMZ.

Examples

```r
## Not run:
library(msdata)
mzdatapath <- system.file("iontrap", package = "msdata")
mzdatafiles<-list.files(mzdatapath, pattern = "extracted.mzData", recursive = TRUE, full.names = TRUE)
xs <- xcmsSet(mzdatafiles, method = "MS1")
# takes only one file from the file set
xfrag <- xcmsFragments(xs)
found<-findneutral(xfrag, 58.1455, 50)
## End(Not run)
```
findPeaks-methods  Feature detection for GC/MS and LC/MS Data - methods

Description

A number of peak pickers exist in XCMS. `findPeaks` is the generic method.

Arguments

- `object`  
  xcmsRaw-class object

- `method`  
  Method to use for peak detection. See details.

- `...`  
  Optional arguments to be passed along

Details

Different algorithms can be used by specifying them with the `method` argument. For example to use the matched filter approach described by Smith et al (2006) one would use: `findPeaks(object, method="matchedFilter")`. This is also the default.

Further arguments given by `...` are passed through to the function implementing the method.

A character vector of `nicknames` for the algorithms available is returned by `getOption("BioC")$xcms$findPeaks.methods`. If the nickname of a method is called "centWave", the help page for that specific method can be accessed with `?findPeaks.centWave`.

Value

A matrix with columns:

- `mz`  
  weighted (by intensity) mean of peak m/z across scans

- `mzmin`  
  m/z of minimum step

- `mzmax`  
  m/z of maximum step

- `rt`  
  retention time of peak midpoint

- `rtmin`  
  leading edge of peak retention time

- `rtmax`  
  trailing edge of peak retention time

- `into`  
  integrated area of original (raw) peak

- `maxo`  
  maximum intensity of original (raw) peak

and additional columns depending on the choosen method.

Methods

`object = "xcmsRaw"`  
`findPeaks(object, ...)`

See Also

- `findPeaks.matchedFilter`
- `findPeaks.centWave`
- `findPeaks.addPredictedIsotopeFeatures`
- `findPeaks.centWaveWithPredictedIsotopeROIs`
- `xcmsRaw-class`
findPeaks.addPredictedIsotopeFeatures-methods

Feature detection based on predicted isotope features for high resolution LC/MS data

Description

Peak density and wavelet based feature detection aiming at isotope peaks for high resolution LC/MS data in centroid mode

Arguments

object xcmsSet object
ppm maximal tolerated m/z deviation in consecutive scans, in ppm (parts per million)
peakwidth Chromatographic peak width, given as range (min,max) in seconds
prefilter prefilter=c(k,I). Prefilter step for the first phase. Mass traces are only retained if they contain at least k peaks with intensity >= I.
mzCenterFun Function to calculate the m/z center of the feature: \( \mu_{\text{mean}} \) intensity weighted mean of the feature m/z values, \( \mu_{\text{mean}} \) mean of the feature m/z values, apex use m/z value at peak apex, \( \mu_{\text{meanApex}} \) intensity weighted mean of the m/z value at peak apex and the m/z value left and right of it, \( \mu_{\text{meanApex}} \) mean of the m/z value at peak apex and the m/z value left and right of it.
integrate Integration method. If =1 peak limits are found through descent on the mexican hat filtered data, if =2 the descent is done on the real data. Method 2 is very accurate but prone to noise, while method 1 is more robust to noise but less exact.
mzdifff minimum difference in m/z for peaks with overlapping retention times, can be negative to allow overlap
fitgauss logical, if TRUE a Gaussian is fitted to each peak
scrange optional argument which is useful for data that was centroided without any intensity threshold, centroids with intensity < noise are omitted from ROI detection
sleep number of seconds to pause between plotting peak finding cycles
verbose.columns logical, if TRUE additional peak meta data columns are returned
xcmsPeaks peak list picked using the centWave algorithm with parameter verbose.columns set to TRUE (columns scmin and scmax needed)
snthresh signal to noise ratio cutoff, definition see below.
maxcharge max. number of the isotope charge.
maxiso max. number of the isotope peaks to predict for each detected feature.
mzIntervalExtension logical, if TRUE predicted isotope ROIs (regions of interest) are extended in the m/z dimension to increase the detection of low intensity and hence noisy peaks.
Details

This algorithm is most suitable for high resolution LC/TOF,OrbiTrap,FTICR-MS data in centroid mode. In the first phase of the method isotope ROIs (regions of interest) in the LC/MS map are predicted. In the second phase these mass traces are further analysed. Continuous wavelet transform (CWT) is used to locate chromatographic peaks on different scales. The resulting peak list and the given peak list (xcmsPeaks) are merged and redundant peaks are removed.

Value

A matrix with columns:

- \( m_z \): weighted (by intensity) mean of peak m/z across scans
- \( m_{z\text{min}} \): m/z peak minimum
- \( m_{z\text{max}} \): m/z peak maximum
- \( r_t \): retention time of peak midpoint
- \( r_{t\text{min}} \): leading edge of peak retention time
- \( r_{t\text{max}} \): trailing edge of peak retention time
- \( \text{int} \): integrated peak intensity
- \( \text{intb} \): baseline corrected integrated peak intensity
- \( \text{maxo} \): maximum peak intensity
- \( \text{sn} \): Signal/Noise ratio, defined as \( \frac{\text{maxo} - \text{baseline}}{\text{sd}} \), where \( \text{maxo} \) is the maximum peak intensity, \( \text{baseline} \) the estimated baseline value and \( \text{sd} \) the standard deviation of local chromatographic noise.
- \( \text{egauss} \): RMSE of Gaussian fit
  
  if verbose.columns is TRUE additionally:
  
  - \( \mu \): Gaussian parameter \( \mu \)
  - \( \sigma \): Gaussian parameter \( \sigma \)
  - \( h \): Gaussian parameter \( h \)
  - \( f \): Region number of m/z ROI where the peak was localised
  - \( \text{dppm} \): m/z deviation of mass trace across scans in ppm
  - \( \text{scale} \): Scale on which the peak was localised
  - \( \text{scpos} \): Peak position found by wavelet analysis
  - \( \text{scmin} \): Left peak limit found by wavelet analysis (scan number)
  - \( \text{scmax} \): Right peak limit found by wavelet analysis (scan number)

Methods

\[
\text{object} = "\text{xcmsRaw}"
\]

findPeaks.centWave(object, ppm=25, peakwidth=c(20,50), prefilter=c(3,100), ...

Author(s)

Ralf Tautenhahn
findPeaks.centWave-methods

References


See Also

findPeaks.centWave findPeaks-methods xcmsRaw-class

findPeaks.centWave-methods

Feature detection for high resolution LC/MS data

Description

Peak density and wavelet based feature detection for high resolution LC/MS data in centroid mode

Arguments

- **object**: xcmsSet object
- **ppm**: maxmial tolerated m/z deviation in consecutive scans, in ppm (parts per million)
- **peakwidth**: Chromatographic peak width, given as range (min,max) in seconds
- **snthresh**: signal to noise ratio cutoff, definition see below.
- **prefilter**: prefilter=c(k,I). Prefilter step for the first phase. Mass traces are only retained if they contain at least k peaks with intensity >= I.
- **mzCenterFun**: Function to calculate the m/z center of the feature: wMean intensity weighted mean of the feature m/z values, mean mean of the feature m/z values, apex use m/z value at peak apex, wMeanApex3 intensity weighted mean of the m/z value at peak apex and the m/z value left and right of it, meanApex3 mean of the m/z value at peak apex and the m/z value left and right of it.
- **integrate**: Integration method. If =1 peak limits are found through descent on the mexican hat filtered data, if =2 the descent is done on the real data. Method 2 is very accurate but prone to noise, while method 1 is more robust to noise but less exact.
- **mzdiff**: minimum difference in m/z for peaks with overlapping retention times, can be negative to allow overlap
- **fitgauss**: logical, if TRUE a Gaussian is fitted to each peak
- **scanrange**: scan range to process
- **noise**: optional argument which is useful for data that was centroided without any intensity threshold, centroids with intensity < noise are omitted from ROI detection
- **sleep**: number of seconds to pause between plotting peak finding cycles
- **verbose.columns**: logical, if TRUE additional peak meta data columns are returned
ROI.list

A optional list of ROIs that represents detected mass traces (ROIs). If this list is empty (default) then centWave detects the mass trace ROIs, otherwise this step is skipped and the supplied ROIs are used in the peak detection phase. Each ROI object in the list has the following slots: scmin start scan index, scmax end scan index, mzmin minimum m/z, mzmax maximum m/z, length number of scans, intensity summed intensity.

firstBaselineCheck

logical, if TRUE continuous data within ROI is checked to be above 1st baseline

roiScales

numeric, optional vector of scales for each ROI in ROI.list to be used for the centWave-wavelets

Details

This algorithm is most suitable for high resolution LC/(TOF,OrbiTrap,FTICR)-MS data in centroid mode. In the first phase of the method mass traces (characterised as regions with less than ppm m/z deviation in consecutive scans) in the LC/MS map are located. In the second phase these mass traces are further analysed. Continuous wavelet transform (CWT) is used to locate chromatographic peaks on different scales.

Value

A matrix with columns:

mz
weighted (by intensity) mean of peak m/z across scans
mzmin
m/z peak minimum
mzmax
m/z peak maximum
rt
retention time of peak midpoint
rtmin
leading edge of peak retention time
rtmax
trailing edge of peak retention time
into
integrated peak intensity
intb
baseline corrected integrated peak intensity
maxo
maximum peak intensity
sn
Signal/Noise ratio, defined as (maxo - baseline)/sd, where maxo is the maximum peak intensity, baseline the estimated baseline value and sd the standard deviation of local chromatographic noise.
e gauss
RMSE of Gaussian fit
if verbose.columns is TRUE additionally :
mu
Gaussian parameter mu
sigma
Gaussian parameter sigma
h
Gaussian parameter h
f
Region number of m/z ROI where the peak was localised
dppm
m/z deviation of mass trace across scans in ppm
scale
Scale on which the peak was localised
scpos
Peak position found by wavelet analysis
scmin
Left peak limit found by wavelet analysis (scan number)
scmax
Right peak limit found by wavelet analysis (scan number)
Methods

```r
object = "xcmsRaw" findPeaks.centWave(object, ppm=25, peakwidth=c(20,50), snthresh=10, prefilter=c(3,100), mzCenterFun="wMean", integrate=1, ... scanrange= numeric(), noise=0, sleep=0, verbose.columns=FALSE, ROI.list=list()), firstBaselineCheck=TRUE, roiScales=NULL
```

Author(s)

Ralf Tautenhahn

References

Ralf Tautenhahn, Christoph Böttcher, and Steffen Neumann "Highly sensitive feature detection for high resolution LC/MS" BMC Bioinformatics 2008, 9:504

See Also

findPeaks-methods xcmsRaw-class

---

**Description**

Peak density and wavelet based feature detection for high resolution LC/MS data in centroid mode with additional peak picking of isotope features on basis of isotope peak predictions

**Arguments**

- `object` xcmsSet object
- `ppm` maximal tolerated m/z deviation in consecutive scans, in ppm (parts per million)
- `peakwidth` Chromatographic peak width, given as range (min,max) in seconds
- `snthresh` signal to noise ratio cutoff, definition see below.
- `prefilter` prefilter=c(k,I). Prefilter step for the first phase. Mass traces are only retained if they contain at least k peaks with intensity >= I.
- `mzCenterFun` Function to calculate the m/z center of the feature: wMean intensity weighted mean of the feature m/z values, mean mean of the feature m/z values, apex use m/z value at peak apex, wMeanApex3 intensity weighted mean of the m/z value at peak apex and the m/z value left and right of it, meanApex3 mean of the m/z value at peak apex and the m/z value left and right of it.
- `integrate` Integration method. If =1 peak limits are found through descent on the mexican hat filtered data, if =2 the descent is done on the real data. Method 2 is very accurate but prone to noise, while method 1 is more robust to noise but less exact.
- `mzdiff` minimum difference in m/z for peaks with overlapping retention times, can be negative to allow overlap
- `fitgauss` logical, if TRUE a Gaussian is fitted to each peak
- `scanrange` scan range to process

Feature detection with centWave and additional isotope features
findPeaks.centWaveWithPredictedIsotopeROIs-methods

noise
optional argument which is useful for data that was centroided without any intensity threshold, centroids with intensity < noise are omitted from ROI detection

sleep
number of seconds to pause between plotting peak finding cycles

verbose.columns
logical, if TRUE additional peak meta data columns are returned

ROI.list
A optional list of ROIs that represents detected mass traces (ROIs). If this list is empty (default) then centWave detects the mass trace ROIs, otherwise this step is skipped and the supplied ROIs are used in the peak detection phase. Each ROI object in the list has the following slots: scmin start scan index, scmax end scan index, mzmin minimum m/z, mzmax maximum m/z, length number of scans, intensity summed intensity.

firstBaselineCheck
logical, if TRUE continuous data within ROI is checked to be above 1st baseline

roiScales
numeric, optional vector of scales for each ROI in ROI.list to be used for the centWave-wavelets

snthreshIsoROIs
signal to noise ratio cutoff for predicted isotope ROIs, definition see below.

maxcharge
max. number of the isotope charge.

maxiso
max. number of the isotope peaks to predict for each detected feature.

mzIntervalExtension
logical, if TRUE predicted isotope ROIs (regions of interest) are extended in the m/z dimension to increase the detection of low intensity and hence noisy peaks.

Details
This algorithm is most suitable for high resolution LC/{TOF,OrbiTrap,FTICR}-MS data in centroid mode. The centWave algorithm is applied in two peak picking steps as follows. In the first peak picking step ROIs (regions of interest, characterised as regions with less than ppm m/z deviation in consecutive scans) in the LC/MS map are located and further analysed using continuous wavelet transform (CWT) for the localization of chromatographic peaks on different scales. In the second peak picking step isotope ROIs in the LC/MS map are predicted further analysed using continuous wavelet transform (CWT) for the localization of chromatographic peaks on different scales. The peak lists resulting from both peak picking steps are merged and redundant peaks are removed.

Value
A matrix with columns:

mz weighted (by intensity) mean of peak m/z across scans
mzmin m/z peak minimum
mzmax m/z peak maximum
rt retention time of peak midpoint
rtmin leading edge of peak retention time
rtmax trailing edge of peak retention time
into integrated peak intensity
intb baseline corrected integrated peak intensity
maxo maximum peak intensity
Signal/Noise ratio, defined as \((\text{maxo} - \text{baseline})/\text{sd}\), where \text{maxo} is the maximum peak intensity, \text{baseline} the estimated baseline value and \text{sd} the standard deviation of local chromatographic noise.

RMSE of Gaussian fit

Gaussian parameter \(\mu\)

Gaussian parameter \(\sigma\)

Gaussian parameter \(h\)

Region number of m/z ROI where the peak was localised

m/z deviation of mass trace across scans in ppm

Scale on which the peak was localised

Peak position found by wavelet analysis

Left peak limit found by wavelet analysis (scan number)

Right peak limit found by wavelet analysis (scan number)

\textbf{Methods}

\texttt{object = "xcmsRaw"}

\texttt{findPeaks.centWaveWithPredictedIsotopeROIs(object, ppm=25, peakwidth=c(20,50), ... firstBaselineCheck=TRUE, roiScales=NULL, snthreshIsoROIs=6.25, maxcharge=3, maxiso=5, mzIntervalExtension=TRUE)}

\textbf{Author(s)}

Ralf Tautenhahn

\textbf{References}

Ralf Tautenhahn, Christoph Böttcher, and Steffen Neumann "Highly sensitive feature detection for high resolution LC/MS" BMC Bioinformatics 2008, 9:504

Hendrik Treutler and Steffen Neumann. "Prediction, detection, and validation of isotope clusters in mass spectrometry data" Submitted to Metabolites 2016, Special Issue "Bioinformatics and Data Analysis"

\textbf{See Also}

\texttt{findPeaks.addPredictedIsotopeFeatures} \texttt{findPeaks.centWave} \texttt{findPeaks-methods} \texttt{xcmsRaw-class}

\textbf{Description}

Massifquant is a Kalman filter (KF) based feature detection for XC-MS data in centroid mode (currently in experimental stage). Optionally allows for calling the method "centWave" on features discovered by Massifquant to further refine the feature detection; to do so, supply any additional parameters specific to centWave (even more experimental). The method may be conveniently called through the \texttt{xcmsSet(\ldots)} method.
Arguments

The following arguments are specific to Massifquant. Any additional arguments supplied must correspond as specified by the method findPeaks.centWave.

An xcmsRaw object.

**objVal**: Numeric; Suggested values: (0.1-3.0). This setting helps determine the the Kalman Filter prediction margin of error. A real centroid belonging to a bona fide feature must fall within the KF prediction margin of error. Much like in the construction of a confidence interval, criticalVal loosely translates to be a multiplier of the standard error of the prediction reported by the Kalman Filter. If the features in the XC-MS sample have a small mass deviance in ppm error, a smaller critical value might be better and vice versa.

**consecMissedLimit**: Integer; Suggested values: (1,2,3). While a feature is in the process of being detected by a Kalman Filter, the Kalman Filter may not find a predicted centroid in every scan. After 1 or more consecutive failed predictions, this setting informs Massifquant when to stop a Kalman Filter from following a candidate feature.

**prefilter**: Numeric Vector: (Positive Integer, Positive Numeric): The first argument is only used if (withWave = 1); see centWave for details. The second argument specifies the minimum threshold for the maximum intensity of a feature that must be met.

**peakwidth**: Integer Vector: (Positive Integer, Positive Integer): Only the first argument is used for Massifquant, which specifies the minimum feature length in time scans. If centWave is used, then the second argument is the maximum feature length subject to being greater than the minimum feature length.

**ppm**: The minimum estimated parts per million mass resolution a feature must possess.

**unions**: Integer: set to 1 if apply t-test union on segmentation; set to 0 if no t-test to be applied on chromatographically continuous features sharing same m/z range. Explanation: With very few data points, sometimes a Kalman Filter stops tracking a feature prematurely. Another Kalman Filter is instantiated and begins following the rest of the signal. Because tracking is done backwards to forwards, this algorithmic defect leaves a real feature divided into two segments or more. With this option turned on, the program identifies segmented features and combines them (merges them) into one with a two sample t-test. The potential danger of this option is that some truly distinct features may be merged.

**withWave**: Integer: set to 1 if turned on; set to 0 if turned off. Allows the user to find features first with Massifquant and then filter those features with the second phase of centWave, which includes wavelet estimation.

**checkBack**: Integer: set to 1 if turned on; set to 0 if turned off. The convergence of a Kalman Filter to a feature’s precise m/z mapping is very fast, but sometimes it incorporates erroneous centroids as part of a feature (especially early on). The "scan-Back" option is an attempt to remove the occasional outlier that lies beyond the converged bounds of the Kalman Filter. The option does not directly affect identification of a feature because it is a postprocessing measure; it has not shown to be a extremely useful thus far and the default is set to being turned off.

Details

This algorithm’s performance has been tested rigorously on high resolution LC/(OrbiTrap, TOF)-MS data in centroid mode. Simultaneous kalman filters identify features and calculate their area...
findPeaks.massifquant-methods

under the curve. The default parameters are set to operate on a complex LC-MS Orbitrap sample. Users will find it useful to do some simple exploratory data analysis to find out where to set a minimum intensity, and identify how many scans an average feature spans. The "consecMissedLimit" parameter has yielded good performance on Orbitrap data when set to (2) and on TOF data it was found best to be at (1). This may change as the algorithm has yet to be tested on many samples. The "criticalValue" parameter is perhaps most difficult to dial in appropriately and visual inspection of peak identification is the best suggested tool for quick optimization. The "ppm" and "checkBack" parameters have shown less influence than the other parameters and exist to give users flexibility and better accuracy.

Value

If the method findPeaks.massifquant(...) is used, then a matrix is returned with rows corresponding to features, and properties of the features listed with the following column names. Otherwise, if centrWave feature is used also (withWave = 1), or Massifquant is called through the xcmsSet(...) method, then their corresponding return values are used.

- **mz**: weighted m/z mean (weighted by intensity) of the feature
- **mzmin**: m/z lower boundary of the feature
- **mzmax**: m/z upper boundary of the feature
- **rtmin**: starting scan time of the feature
- **rtmax**: starting scan time of the feature
- **into**: the raw quantitation (area under the curve) of the feature.
- **area**: feature area that is not normalized by the scan rate.

Methods

```r
object = "xcmsRaw" findPeaks.massifquant(object, ppm=10, peakwidth=c(20,50), snthresh=10, prefilter=c(3,100), mzCenterFun="wMean", ... sleep=0, verbose.columns=FALSE, criticalValue = 1.125, consecMissedLimit = 2, unions = 1, checkBack = 0, withWave = 0)
```

Author(s)

Christopher Conley

References


See Also

- `findPeaks-methods`
- `xcmsSet`
- `xcmsRaw`
- `xcmsRaw-class`

Examples

```r
library(faahKO)
library(xcms)
# load all the wild type and Knock out samples
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
# run the massifquant analysis
xset <- xcmsSet(cdffiles, method = "massifquant",
```
findPeaks.matchedFilter-methods

Feature detection in the chromatographic time domain

Description

Find peaks in the chromatographic time domain of the profile matrix.

Arguments

- **object**: xcmsRaw object
- **fwhm**: full width at half maximum of matched filtration gaussian model peak. Only used to calculate the actual sigma, see below.
- **sigma**: standard deviation (width) of matched filtration model peak
- **max**: maximum number of peaks per extracted ion chromatogram
- **snthresh**: signal to noise ratio cutoff
- **step**: step size to use for profile generation
- **steps**: number of steps to merge prior to filtration
- **mzdiff**: minimum difference in m/z for peaks with overlapping retention times
- **index**: return indicies instead of values for m/z and retention times
- **sleep**: number of seconds to pause between plotting peak finding cycles
- **scanrange**: scan range to process

Details

The method calculates the profile matrix (i.e. intensities in bins along the M/Z dimension) on the fly using one of the methods described on the profBin help page.

Value

A matrix with columns:

- **mz**: weighted (by intensity) mean of peak m/z across scans
- **mzmin**: m/z of minimum step
- **mzmax**: m/z of maximum step
- **rt**: retention time of peak midpoint
- **rtmin**: leading edge of peak retention time
- **rtmax**: trailing edge of peak retention time
- **into**: integrated area of original (raw) peak
**Methods**

```r
findPeaks.MS1-methods
```

```r
object = "xcmsRaw" findPeaks.matchedFilter(object, fwhm = 30, sigma = fwhm/2.3548, max = 5,
```

**Author(s)**

Colin A. Smith, <csmith@scripps.edu>

**See Also**

`findPeaks-methods xcmsRaw-class`

---

**findPeaks.MS1-methods Collecting MS1 precursor peaks**

**Description**

Collecting Tandem MS or MS^n Mass Spectrometry precursor peaks as annotated in XML raw file

**Arguments**

- `object`: xcmsRaw object

**Details**

Some mass spectrometers can acquire MS1 and MS2 (or MS^n scans) quasi simultaneously, e.g. in data dependent tandem MS or DDIT mode.

Since xcmsFragments attaches all MS^n peaks to MS1 peaks in xcmsSet, it is important that findPeaks and xcmsSet do not miss any MS1 precursor peak.

To be sure that all MS1 precursor peaks are in an xcmsSet, findPeaks.MS1 does not do an actual peak picking, but simply uses the annotation stored in mzXML, mzData or mzML raw files.

This relies on the following XML tags:

- **mzData:**
  ```xml
  <spectrum id="463"> <spectrumInstrument msLevel="2">
  <cvParam cvLabel="psi" accession="PSI:1000039" name="TimeInSeconds" value="92.7743"/>
  </spectrumInstrument> <precursor msLevel="1" spectrumRef="461">
  <cvParam cvLabel="psi" accession="PSI:1000040" name="MassToChargeRatio" value="462.091"/>
  <cvParam cvLabel="psi" accession="PSI:1000042" name="Intensity" value="366.674"/>
  </precursor> </spectrum>
  ```

- **mzXML:**
  ```xml
  <scan num="17" msLevel="2" retentionTime="PT1.5224S"> <precursorMz precursorIntensity="125245">220.1828003</precursorMz>
  </scan>
  ```

Several mzXML and mzData converters are known to create incomplete files, either without intensities (they will be set to 0) or without the precursor retention time (then a reasonably close rt will be chosen. NYI).
findPeaks.MSW-methods

Value
A matrix with columns:

- mz, mzmin, mzmax
  annotated MS1 precursor selection mass
- rt, rtmin, rtmax
  annotated MS1 precursor retention time
- into, maxo, sn
  annotated MS1 precursor intensity

Methods

```r
object = "xcmsRaw" findPeaks.MS1(object)
```

Author(s)
Steffen Neumann, <sneumann@ipb-halle.de>

See Also

- findPeaks-methods xcmsRaw-class

Description
Processing Mass Spectrometry direct-injection spectrum by using wavelet based algorithm.

Arguments

- `object` xcmsSet object
- `snthresh` signal to noise ratio cutoff
- `scales` scales of CWT
- `nearbyPeak` Determine whether to include the nearby small peaks of major peaks. TRUE by default
- `sleep` number of seconds to pause between plotting peak finding cycles
- `verbose.columns` additional peak meta data columns are returned

Details
This is a wrapper around the peak picker in the bioconductor package MassSpecWavelet calling 'cwt', 'get.localMaximum.cwt', 'get.ridge', 'identify.majorPeaks' and tuneIn.peakInfo.
getEIC-methods

Value
A matrix with columns:

- mz: m/z value of the peak at the centroid position
- mzmin: m/z value at the start-point of the peak
- mzmax: m/z value at the end-point of the peak
- rt: always -1
- rtmin: always -1
- rtmax: always -1
- into: integrated area of original (raw) peak
- maxo: intensity of original (raw) peak at the centroid position
- intf: always NA
- maxf: maximum MSW-filter response of the peak
- sn: Signal/Noise ratio

Methods

`object = "xcmsRaw"` findPeaks.MSW(object, snthresh=3, scales=seq(1,22,3), nearbyPeak=TRUE, 

Author(s)
Steffen Neumann, Joachim Kutzer, <sneumann|jkutzer@ipb-halle.de>

See Also
findPeaks-methods xcmsRaw-class peakDetectionCWT

Description
Generate multiple extracted ion chromatograms for m/z values of interest. For xcmsSet objects, reread original raw data and apply precomputed retention time correction, if applicable. Note that this method will always return profile, not raw data (with profile data being the binned data along M/Z). See details for further information.

Arguments

- `object`: the xcmsRaw or xcmsSet object
- `mzrange`: Either a two column matrix with minimum or maximum m/z or a matrix of any dimensions containing columns mzmin and mzmax. If not specified, the method for xcmsRaw returns the base peak chromatogram (BPC, i.e. the most intense signal for each RT across all m/z). For xcmsSet objects the group data will be used if mzrange is not provided.
getPeaks-methods

rtrange  A two column matrix the same size as mzrange with minimum and maximum retention times between which to return EIC data points. If not specified, the method returns the chromatogram for the full RT range. For xcmsSet objects, it may also be a single number specifying the time window around the peak to return EIC data points.

step  step (bin) size to use for profile generation. Note that a value of step = 0 is not supported.

groupid  either character vector with names or integer vector with indices of peak groups for which to get EICs.

sampleidx  either character vector with names or integer vector with indices of samples for which to get EICs.

rt  "corrected" for using corrected retention times, or "raw" for using raw retention times.

Details

In contrast to the rawEIC method, that extracts the actual raw values, this method extracts them from the object’s profile matrix (or if the provided step argument does not match the profStep of the object the profile matrix is calculated on the fly and the values returned).

Value

For xcmsSet and xcmsRaw objects, an xcmsEIC object.

Methods

object = "xcmsRaw" getEIC(object, mzrange, rtrange = NULL, step = 0.1)

object = "xcmsSet" getEIC(object, mzrange, rtrange = 200, groupidx, sampleidx = sampname, rt = c("corrected", "raw"))

See Also

xcmsRaw-class, xcmsSet-class, xcmsEIC-class, rawEIC

getPeaks-methods  Get peak intensities for specified regions

Description

Integrate extracted ion chromatograms in pre-defined defined regions. Return output similar to findPeaks.

Arguments

object  the xcmsSet object

peakrange  matrix or data frame with 4 columns: mzmin, mzmax, rtmin, rtmax (they must be in that order or named)

step  step size to use for profile generation
Value

A matrix with columns:

- \(i\): rank of peak identified in merged EIC (\(\leq \text{max}\)), always NA
- \(mz\): weighted (by intensity) mean of peak m/z across scans
- \(mzmin\): m/z of minimum step
- \(mzmax\): m/z of maximum step
- \(ret\): retention time of peak midpoint
- \(retmin\): leading edge of peak retention time
- \(retmax\): trailing edge of peak retention time
- \(into\): integrated area of original (raw) peak
- \(intf\): integrated area of filtered peak, always NA
- \(maxo\): maximum intensity of original (raw) peak
- \(maxf\): maximum intensity of filtered peak, always NA

Methods

For \(\text{object} = \"xcmsRaw\"\):

- \(\text{getPeaks(object, peakrange, step = 0.1)}\)

See Also

- \(\text{xcmsRaw-class}\)
getXcmsRaw-methods

**getSpec-methods**

*Get average m/z and intensity values for multiple mass scans*

**Description**

Return full-resolution averaged data from multiple mass scans.

**Arguments**

- `object` the `xcmsRaw` object
- `...` arguments passed to `profRange` used to specify the spectral segments of interest for averaging

**Details**

Based on the mass points from the spectra selected, a master unique list of masses is generated. Every spectra is interpolated at those masses and then averaged.

**Value**

A matrix with two columns:

- `mz` m/z values
- `intensity` intensity values

**Methods**

```r
object = "xcmsRaw" getSpec(object, ...)
```

**See Also**

`xcmsRaw-class`, `profRange`, `getScan`

---

getXcmsRaw-methods

*Load the raw data for one or more files in the xcmsSet*

**Description**

Reads the raw data applies eventual retention time corrections and waters Lock mass correction and returns it as an `xcmsRaw` object (or list of `xcmsRaw` objects) for one or more files of the `xcmsSet` object.

**Arguments**

- `object` the `xcmsSet` object
- `sampleidx` The index of the sample for which the raw data should be returned. Can be a single number or a numeric vector with the indices. Alternatively, the file name can be specified.
- `profmethod` The profile method.
- `profstep` The profile step.
- `rt` Whether corrected or raw retention times should be returned.
- `...` Additional arguments submitted to the `xcmsRaw` function.
Value
A single xcmsRaw object or a list of xcmsRaw objects.

Methods

\[
\text{object} = "\text{xcmSet}\"
\text{getXcmsRaw(object, sampleidx=1, profmethod=profinfo(object)$method, profstep=...} \)
\]

Author(s)
Johannes Rainer, <johannes.rainer@eurac.edu>

See Also
xcmsRaw-class.

Description
A number of grouping (or alignment) methods exist in XCMS. group is the generic method.

Arguments

\begin{itemize}
\item \textbf{object} \hspace{1cm} \texttt{xcmsSet-class} object
\item \textbf{method} \hspace{1cm} Method to use for grouping. See details.
\item \textbf{...} \hspace{1cm} Optional arguments to be passed along
\end{itemize}

Details
Different algorithms can be used by specifying them with the \texttt{method} argument. For example to use the density-based approach described by Smith et al (2006) one would use: \texttt{group(object, method="density")}. This is also the default.

Further arguments given by \texttt{...} are passed through to the function implementing the method.

A character vector of \texttt{nicknames} for the algorithms available is returned by \texttt{getOption("BioC")$xcms$group.methods}. If the nickname of a method is called "mzClust", the help page for that specific method can be accessed with \texttt{?group.mzClust}.

Value
An xcmsSet object with peak group assignments and statistics.

Methods

\[
\text{object} = "\text{xcmSet}\" \quad \text{group(object, ...)}
\]

See Also

\texttt{group.density group.mzClust group.nearest xcmsSet-class,}
group.density

**Group peaks from different samples together**

**Description**

Group peaks together across samples using overlapping m/z bins and calculation of smoothed peak distributions in chromatographic time.

**Arguments**

- **object** the `xcmsSet` object
- **minfrac** minimum fraction of samples necessary in at least one of the sample groups for it to be a valid group
- **minsamp** minimum number of samples necessary in at least one of the sample groups for it to be a valid group
- **bw** bandwidth (standard deviation or half width at half maximum) of gaussian smoothing kernel to apply to the peak density chromatogram
- **mzwid** width of overlapping m/z slices to use for creating peak density chromatograms and grouping peaks across samples
- **max** maximum number of groups to identify in a single m/z slice
- **sleep** seconds to pause between plotting successive steps of the peak grouping algorithm. peaks are plotted as points showing relative intensity. identified groups are flanked by dotted vertical lines.

**Value**

An `xcmsSet` object with peak group assignments and statistics.

**Methods**

```r
object = "xcmsSet"    group(object, bw = 30, minfrac = 0.5, minsamp = 1, mzwid = 0.25, max = 50, sleep = 0)
```

**See Also**

`xcmsSet-class`, `density`

---

group.mzClust

**Group Peaks via High Resolution Alignment**

**Description**

Runs high resolution alignment on single spectra samples stored in a given `xcmsSet`. 
**group.nearest**

**Group peaks from different samples together**

**Description**

Group peaks together across samples by creating a master peak list and assigning corresponding peaks from all samples. It is inspired by the alignment algorithm of mzMine. For further details check [http://mzmine.sourceforge.net/](http://mzmine.sourceforge.net/) and


**Arguments**

- **object**: a xcmsSet with peaks
- **mzppm**: the relative error used for clustering/grouping in ppm (parts per million)
- **mzabs**: the absolute error used for clustering/grouping
- **minsamp**: set the minimum number of samples in one bin
- **minfrac**: set the minimum fraction of each class in one bin

**Value**

Returns a xcmsSet with slots groups and groupindex set.

**Methods**

- **object = "xcmsSet"**

  ```r
  group(object, method="mzClust", mzppm = 20, mzabs = 0, minsamp = 1, minfrac = 0)
  ```

**References**

Saira A. Kazmi, Samiran Ghosh, Dong-Guk Shin, Dennis W. Hill and David F. Grant

*Alignment of high resolution mass spectra: development of a heuristic approach for metabolomics.*


**See Also**

- [xcmsSet-class](#)

**Examples**

```r
## Not run:
library(msdata)
mzdatapath <- system.file("fticr", package = "msdata")
mzdatafiles <- list.files(mzdatapath, recursive = TRUE, full.names = TRUE)
xs <- xcmsSet(method="MSW", files=mzdatafiles, scales=c(1,7), SNR.method="data.mean", winSize.noise=500, peakThr=80000, amp.Th=0.005)
xsg <- group(xs, method="mzClust")
## End(Not run)
```
Arguments

- **object**: the xcmsSet object
- **mzVsRTbalance**: Multiplicator for mz value before calculating the (euclidean) distance between two peaks.
- **mzCheck**: Maximum tolerated distance for mz.
- **rtCheck**: Maximum tolerated distance for RT.
- **kNN**: Number of nearest Neighbours to check

Value

An xcmsSet object with peak group assignments and statistics.

Methods

```r
object = "xcmsSet"  group(object, mzVsRTbalance=10, mzCheck=0.2, rtCheck=15, kNN=10)
```

See Also

`xcmsSet-class`, `group.density` and `group.mzClust`

Examples

```r
## Not run: library(xcms)
library(faahKO)  ## These files do not have this problem to correct for but just for an example
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)

xset<-xcmsSet(cdffiles)
gxset<-group(xset, method="nearest")
  ## this is the same as  
  # gxset<-group.nearest(xset)
nrow(gxset@groups) == 1096  ## the number of features before minFrac

post.minFrac<function(object, minFrac=0.5){
  ix.minFrac<-sapply(1:length(unique(sampclass(object))), function(x, object, mf){
    meta<-groups(object)
    minFrac.idx<-numeric(length=nrow(meta))
    idx<-which(meta[,levels(sampclass(object))[x]] >= mf*length(which(levels(sampclass(object))[x] == sampclass(object))))
    minFrac.idx[idx]<-1
    return(minFrac.idx)
  }, object, minFrac)
  ix.minFrac<as.logical(apply(ix.minFrac, 1, sum))
  ix<-which(ix.minFrac == TRUE)
  return(ix)
}

## using the above function we can get a post processing minFrac
ix<-post.minFrac(gxset)

gxset.post<-gxset  ## copy the xcmsSet object
gxset.post@groupidx<-gxset@groupidx[ix]
gxset.post@groups<-gxset@groups[ix,]
```
## groupnames-methods

### Generate unique names for peak groups

### Description

Allow linking of peak group data between classes using unique group names that remain the same as long as no re-grouping occurs.

### Arguments

- **object**: the `xcmsSet` or `xcmsEIC` object
- **mzdec**: number of decimal places to use for m/z
- **rtdec**: number of decimal places to use for retention time
- **template**: a character vector with existing group names whose format should be emulated

### Value

A character vector with unique names for each peak group in the object. The format is M[m/z]T[time in seconds].

### Methods

- `object = "xcmsSet"` (object, mzdec = 0, rtdec = 0, template = NULL)
- `object = "xcmsEIC"` (object)

### See Also

- `xcmsSet-class`, `xcmsEIC-class`

---

## groupval-methods

### Extract a matrix of peak values for each group

### Description

Generate a matrix of peak values with rows for every group and columns for every sample. The value included in the matrix can be any of the columns from the `xcmsSet` peaks slot matrix. Collisions where more than one peak from a single sample are in the same group get resolved with one of several user-selectable methods.

### Arguments

- **object**: the `xcmsSet` object
- **method**: conflict resolution method, "medret" to use the peak closest to the median retention time or "maxint" to use the peak with the highest intensity
- **value**: name of peak column to enter into returned matrix, or "index" for index to the corresponding row in the peaks slot matrix
- **intensity**: if method == "maxint", name of peak column to use for intensity
Value

A matrix with rows for every group and columns for every sample. Missing peaks have NA values.

Methods

\[
\text{object = "xCMS\Set" groupval(object, method = c("medret", "maxint"), value = "index", intensity = "into")}
\]

See Also

\text{xCMS\Set-class}

---

**image-methods**

**Plot log intensity image of a xCMS\Raw object**

Description

Create log intensity false-color image of a xCMS\Raw object plotted with m/z and retention time axes

Arguments

- \text{x} xCMS\Raw object
- \text{col} vector of colors to use for the image
- \text{...} arguments for \text{profRange}

Methods

\[
\text{x = "xCMSRaw" image(x, col = rainbow(256), ...)}
\]

Author(s)

Colin A. Smith, <csmith@scripps.edu>

See Also

\text{xCMSRaw-class}
levelplot-methods

Plot log intensity image of a `xcmsRaw` object

Description
Create an image of the raw (profile) data m/z against retention time, with the intensity color coded.

Arguments
- **x**: `xcmsRaw` object.
- **log**: Whether the intensity should be log transformed.
- **col.regions**: The color ramp that should be used for encoding of the intensity.
- **rt**: Whether the original (`rt="raw"`) or the corrected (`rt="corrected"`) retention times should be used.
- **...**: Arguments for `profRange`.

Methods
- **`x = "xcmsRaw"`**: `levelplot(x, log=TRUE, col.regions=colorRampPalette(brewer.pal(9, "YlOrRd"))(256), ...)`
- **`x = "xcmsSet"`**: `levelplot(x, log=TRUE, col.regions=colorRampPalette(brewer.pal(9, "YlOrRd"))(256), rt="raw", ...)`

Author(s)
Johannes Rainer, <johannes.rainer@eurac.edu>

See Also
- `xcmsRaw-class`, `xcmsSet-class`

loadRaw-methods

Read binary data from a source

Description
This function extracts the raw data which will be used an `xcmsRaw` object. Further processing of data is done in the `xcmsRaw` constructor.

Arguments
- **object**: Specification of a data source (such as a file name or database query)

Details
The implementing methods decide how to gather the data.
Value

A list containing elements describing the data source. The `rt`, `scanindex`, `tic`, and `acquisitionNum` components each have one entry per scan. They are "parallel" in the sense that `rt[1]`, `scanindex[1]`, and `acquisitionNum[1]` all refer to the same scan. The list containst he following components:

- **rt**` Numeric vector with acquisition time (in seconds) for each scan
- **tic**` Numeric vector with Total Ion Count for each scan
- **scanindex**` Integer vector with starting positions of each scan in the `mz` and `intensity` components. It is an exclusive offset, so `scanindex[i]` is the offset in `mz` and `intensity` before the beginning of scan `i`. This means that the `mz` (respectively `intensity`) values for scan `i` would be from `scanindex[i] + 1` to `scanindex[i + 1]`
- **mz**` Concatenated vector of m/z values for all scans
- **intensity**` Concatenated vector of intensity values for all scans

Methods

signature(object = "xcmsSource") Uses `loadRaw, xcmsSource-method` to extract raw data. Subclasses of `xcmsSource` can provide different ways of fetching data.

Author(s)

Daniel Hackney, <dan@haxney.org>

See Also

`xcmsRaw-class, xcmsSource`

---

**medianFilter**

Apply a median filter to a matrix

Description

For each element in a matrix, replace it with the median of the values around it.

Usage

```r
medianFilter(x, mrad, nrad)
```

Arguments

- **x**` numeric matrix to median filter
- **mrad**` number of rows on either side of the value to use for median calculation
- **nrad**` number of rows on either side of the value to use for median calculation

Value

A matrix whose values have been median filtered
Author(s)

Colin A. Smith, <csmith@scripps.edu>

Examples

```r
mat <- matrix(1:25, nrow=5)
mat
medianFilter(mat, 1, 1)
```

Description

The MS2 and MSn data is stored in separate slots, and can not directly be used by e.g. `findPeaks()`. `msn2xcmsRaw()` will copy the MSn spectra into the "normal" `xcmsRaw` slots.

Usage

```r
msn2xcmsRaw(xmsn)
```

Arguments

- `xmsn` an object of class `xcmsRaw` that contains spectra read with `includeMSn=TRUE`

Details

The default gap value is determined from the 90th percentile of the pair-wise differences between adjacent mass values.

Value

An `xcmsRaw` object

Author(s)

Steffen Neumann <sneumann@ipb-halle.de>

See Also

`xcmsRaw`

Examples

```r
msnfile <- system.file("microtofq/MSMSpos20_6.mzML", package = "msdata")
xrmsn <- xcmsRaw(msnfile, includeMSn=TRUE)
xr <- msn2xcmsRaw(xrmsn)
p <- findPeaks(xr, method="centWave")
```
peakPlots-methods  Plot a grid of a large number of peaks

**Description**

Plot extracted ion chromatograms for many peaks simultaneously, indicating peak integration start and end points with vertical grey lines.

**Arguments**

- **object**: the `xcmsRaw` object
- **peaks**: matrix with peak information as produced by `findPeaks`
- **figs**: two-element vector describing the number of rows and the number of columns of peaks to plot, if missing then an approximately square grid that will fit the number of peaks supplied
- **width**: width of chromatogram retention time to plot for each peak

**Details**

This function is intended to help graphically analyze the results of peak picking. It can help estimate the number of false positives and improper integration start and end points. Its output is very compact and tries to waste as little space as possible. Each plot is labeled with rounded m/z and retention time separated by a space.

**Methods**

- `signature(object = "xcmsSet") plotPeaks(object, peaks, figs, width = 200)`

**See Also**

`xcmsRaw-class, findPeaks, split.screen`

peakTable-methods  Create report of aligned peak intensities

**Description**

Create a report showing all aligned peaks.

**Arguments**

- **object**: the `xcmsSet` object
- **filebase**: base file name to save report. `.tsv` file and `_eic` will be appended to this name for the tabular report and EIC directory, respectively. If blank nothing will be saved
- **...**: arguments passed down to `groupval`, which provides the actual intensities.
Details

This method handles creation of summary reports similar to `diffreport`. It returns a summary report that can optionally be written out to a tab-separated file.

If a base file name is provided, the report (see Value section) will be saved to a tab separated file.

Value

A data frame with the following columns:

- `mz`: median m/z of peaks in the group
- `mzmin`: minimum m/z of peaks in the group
- `mzmax`: maximum m/z of peaks in the group
- `rt`: median retention time of peaks in the group
- `rtmin`: minimum retention time of peaks in the group
- `rtmax`: maximum retention time of peaks in the group
- `npeaks`: number of peaks assigned to the group
- `Sample Classes`: number samples from each sample class represented in the group
- `Sample Names`: integrated intensity value for every sample

Methods

```r
object = "xcmsSet" peakTable(object, filebase = character(), ...)
```

See Also

`xcmsSet-class`.

Examples

```r
## Not run:
library(faahKO)
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
xs<-xcmsSet(cdf files)
xs<-group(xs)
peakTable(xs, filebase="peakList")
## End(Not run)
```
plot.xcmsEIC  
Plot extracted ion chromatograms from multiple files

Description

Batch plot a list of extracted ion chromatograms to the current graphics device.

Arguments

- `x` the `xcmsEIC` object
- `y` optional `xcmsSet` object with peak integration data
- `groupid` either character vector with names or integer vector with indicies of peak groups for which to plot EICs
- `sampleidx` either character vector with names or integer vector with indicies of samples for which to plot EICs
- `rtrange` a two column matrix with minimum and maximum retention times between which to return EIC data points
  - if it has the same number of rows as the number groups in the `xcmsEIC` object, then `sampleidx` is used to subset it. otherwise, it is repeated over the length of `sampleidx`
  - it may also be a single number specifying the time window around the peak for which to plot EIC data
- `col` color to use for plotting extracted ion chromatograms. if missing and `y` is specified, colors are taken from `unclass(sampclass(y))` and the default palette
  - if it is the same length as the number groups in the `xcmsEIC` object, then `sampleidx` is used to subset it. otherwise, it is repeated over the length of `sampleidx`
- `legtext` text to use for legend. if `NULL` and `y` is specified, legend text is taken from the sample class information found in the `xcmsSet`
- `peakint` logical, plot integrated peak area with darkened lines (requires that `y` also be specified)
- `sleep` seconds to pause between plotting EICs
- `...` other graphical parameters

Value

A `xcmsSet` object.

Methods

- `x = "xcmsEIC"`  
  `plot.xcmsEIC(x, y, groupidx = groupnames(x), sampleidx = sampnames(x), rtrange = x@rtrange, col = rep(1, length(sampleidx)), legtext = NULL, peakint = TRUE, sleep = 0, ...)`

Author(s)

Colin A. Smith, <csmith@scripps.edu>

See Also

- `xcmsEIC-class`, `png`, `pdf`, `postscript`
### plotChrom-methods

Plot extracted ion chromatograms from the profile matrix

**Description**

Uses the pre-generated profile mode matrix to plot averaged or base peak extracted ion chromatograms over a specified mass range.

**Arguments**

- `object`: the xcmsRaw object
- `base`: logical, plot a base-peak chromatogram
- `ident`: logical, use mouse to identify and label peaks
- `fitgauss`: logical, fit a gaussian to the largest peak
- `vline`: numeric vector with locations of vertical lines
- `...`: arguments passed to `profRange`

**Value**

If `ident == TRUE`, an integer vector with the indices of the points that were identified. If `fitgauss == TRUE`, a `nls` model with the fitted gaussian. Otherwise a two-column matrix with the plotted points.

**Methods**

```r
object = "xcmsRaw" plotChrom(object, base = FALSE, ident = FALSE, fitgauss = FALSE, vline = numeric(0), ...)
```

**See Also**

- `xcmsRaw-class`

### plotEIC-methods

Plot extracted ion chromatograms for specified m/z range

**Description**

Plot extracted ion chromatogram for m/z values of interest. The raw data is used in contrast to `plotChrom` which uses data from the profile matrix.

**Arguments**

- `object`: xcmsRaw object
- `mzrange`: m/z range for EIC. Uses the full m/z range by default.
- `rtrange`: retention time range for EIC. Uses the full retention time range by default.
- `scanrange`: scan range for EIC
- `mzdec`: Number of decimal places of title m/z values in the eic plot.
- `type`: Specifies how the data should be plotted (by default as a line).
- `add`: If the EIC should be added to an existing plot.
- `...`: Additional parameters passed to the plotting function (e.g. `col` etc).
Value

A two-column matrix with the plotted points.

Methods

object = "xcmsRaw"  
plotEIC(object, mzrange = numeric(), rrange = numeric(), scanrange = numeric(), ...)

Author(s)

Ralf Tautenhahn

See Also

rawEIC, xcmsRaw-class

Description

Plot extracted ion chromatograms for many peaks simultaneously, indicating peak integration start and end points with vertical grey lines.

Arguments

object       the xcmsRaw object
peaks        matrix with peak information as produced by findPeaks
figs         two-element vector describing the number of rows and the number of columns of peaks to plot, if missing then an approximately square grid that will fit the number of peaks supplied
width        width of chromatogram retention time to plot for each peak

Details

This function is intended to help graphically analyze the results of peak picking. It can help estimate the number of false positives and improper integration start and end points. Its output is very compact and tries to waste as little space as possible. Each plot is labeled with rounded m/z and retention time separated by a space.

Methods

object = "xcmsRaw"  plotPeaks(object, peaks, figs, width = 200)

See Also

xcmsRaw-class, findPeaks, split.screen
plotQC

Plot m/z and RT deviations for QC purposes without external reference data

Description

Use "democracy" to determine the average m/z and RT deviations for a grouped xcmsSet, and dependency on sample or absolute m/z

Usage

plotQC(object, sampNames, sampColors, sampOrder, what)

Arguments

- object: A grouped xcmsSet
- sampNames: Override sample names (e.g. with simplified names)
- sampColors: Provide a set of colors (default: monochrome ?)
- sampOrder: Override the order of samples, e.g. to bring them in order of measurement to detect time drift
- what: A vector of which QC plots to generate. "mzdevhist": histogram of m/z deviations. Should be gaussian shaped. If it is multimodal, then some peaks seem to have a systematically higher m/z deviation "rtdevhist": histogram of RT deviations. Should be gaussian shaped. If it is multimodal, then some peaks seem to have a systematically higher RT deviation "mzdevmass": Shows whether m/z deviations are absolute m/z dependent, could indicate miscalibration "mzdev-time": Shows whether m/z deviations are RT dependent, could indicate instrument drift "mzdevsample": median m/z deviation for each sample, indicates outliers "rtdevsample": median RT deviation for each sample, indicates outliers

Details

plotQC() is a wrapper to create a set of diagnostic plots. For the m/z deviations, the median of all m/z within one group are assumed.

Value

No return value

Author(s)

Michael Wenk, Michael Wenk <michael.wenk@student.uni-halle.de>

Examples

library(faahKO)
xsg <- group(faahko)

plotQC(xsg, what="mzdevhist")
plotQC(xsg, what="rtdevhist")
plotQC(xsg, what="mzdevmass")
plotQC(xsg, what="mzdevtime")
plotQC(xsg, what="mzdevsample")
plotQC(xsg, what="rtdevsample")

---

**plotRaw-methods**  
*Scatterplot of raw data points*

**Description**

Produce a scatterplot showing raw data point location in retention time and m/z. This plot is more useful for centroided data than continuum data.

**Arguments**

- **object**: the xcmsRaw object
- **mzrange**: numeric vector of length >= 2 whose range will be used to select the masses to plot
- **rtrange**: numeric vector of length >= 2 whose range will be used to select the retention times to plot
- **scanrange**: numeric vector of length >= 2 whose range will be used to select scans to plot
- **log**: logical, log transform intensity
- **title**: main title of the plot

**Value**

A matrix with the points plotted.

**Methods**

```r
object = "xcmsRaw" plotRaw(object, mzrange = numeric(), rtrange = numeric(), scanrange = numeric(), log=FALSE, title=quote("Var\nRaw Data\nVar"))
```

**See Also**

- xcmsRaw-class

---

**plotrt-methods**  
*Plot retention time deviation profiles*

**Description**

Use corrected retention times for each sample to calculate retention time deviation profiles and plot each on the same graph.
**plotScan-methods**

**Arguments**

- **object** the xcmsSet object
- **col** vector of colors for plotting each sample
- **ty** vector of line and point types for plotting each sample
- **leg** logical plot legend with sample labels
- **densplit** logical, also plot peak overall peak density

**Methods**

- **object = "xcmsSet"** plotrt(object, col = NULL, ty = NULL, leg = TRUE, densplit = FALSE)

**See Also**

- xcmsSet-class, retcor

**plotScan-methods**

*Plot a single mass scan*

**Description**

Plot a single mass scan using the impulse representation. Most useful for centroided data.

**Arguments**

- **object** the xcmsRaw object
- **scan** integer with number of scan to plot
- **mzrange** numeric vector of length $\geq 2$ whose range will be used to select masses to plot
- **ident** logical, use mouse to interactively identify and label individual masses

**Methods**

- **object = "xcmsRaw"** plotScan(object, scan, mzrange = numeric(), ident = FALSE)

**See Also**

- xcmsRaw-class
plotSpec-methods

Plot mass spectra from the profile matrix

Description

Uses the pre-generated profile mode matrix to plot mass spectra over a specified retention time range.

Arguments

- **object**: the `xcmsRaw` object
- **ident**: logical, use mouse to identify and label peaks
- **vline**: numeric vector with locations of vertical lines
- **...**: arguments passed to `profRange`

Value

If `ident == TRUE`, an integer vector with the indecies of the points that were identified. Otherwise a two-column matrix with the plotted points.

Methods

- **object = "xcmsRaw"** plotSpec(object, ident = FALSE, vline = numeric(0), ...)

See Also

- `xcmsRaw-class`

plotSurf-methods

Plot profile matrix 3D surface using OpenGL

Description

This method uses the rgl package to create interactive three dimensonal representations of the profile matrix. It uses the terrain color scheme.

Arguments

- **object**: the `xcmsRaw` object
- **log**: logical, log transform intensity
- **aspect**: numeric vector with aspect ratio of the m/z, retention time and intensity components of the plot
- **...**: arguments passed to `profRange`
plotTIC-methods

Details

The rgl package is still in development and imposes some limitations on the output format. A bug in the axis label code means that the axis labels only go from 0 to the aspect ratio constant of that axis. Additionally the axes are not labeled with what they are.

It is important to only plot a small portion of the profile matrix. Large portions can quickly overwhelm your CPU and memory.

Methods

object = "xcmsRaw" plotSurf(object, log = FALSE, aspect = c(1, 1, .5), ...)

See Also

xcmsRaw-class

plotTIC-methods  Plot total ion count

Description

Plot chromatogram of total ion count. Optionally allow identification of target peaks and viewing/identification of individual spectra.

Arguments

object the xcmsRaw object
ident logical, use mouse to identify and label chromatographic peaks
msident logical, use mouse to identify and label spectral peaks

Value

If ident == TRUE, an integer vector with the indecies of the points that were identified. Otherwise a two-column matrix with the plotted points.

Methods

object = "xcmsRaw" plotTIC(object, ident = FALSE, msident = FALSE)

See Also

xcmsRaw-class
profMedFilt-methods  
Median filtering of the profile matrix

Description

Apply a median filter of given size to a profile matrix.

Arguments

- **object**: the `xcmsRaw` object
- **massrad**: number of m/z grid points on either side to use for median calculation
- **scanrad**: number of scan grid points on either side to use for median calculation

Methods

- object = "xcmsRaw" profMedFilt(object, massrad = 0, scanrad = 0)

See Also

- `xcmsRaw-class`, `medianFilter`

profMethod-methods  
Get and set method for generating profile data

Description

These methods get and set the method for generating profile (matrix) data from raw mass spectral data. It can currently be `bin`, `binlin`, `binlinbase`, or `intlin`.

Methods

- object = "xcmsRaw" profMethod(object)

See Also

- `xcmsRaw-class`, `profMethod`, `profBin`, `plotSpec`, `plotChrom`, `findPeaks`
profRange-methods

Specify a subset of profile mode data

Description

Specify a subset of the profile mode matrix given a mass, time, or scan range. Allow flexible user entry for other functions.

Arguments

- **object**: the `xcmsRaw` object
- **mzrange**: single numeric mass or vector of masses
- **rtrange**: single numeric time (in seconds) or vector of times
- **scanrange**: single integer scan index or vector of indecies
- **...**: arguments to other functions

Details

This function handles selection of mass/time subsets of the profile matrix for other functions. It allows the user to specify such subsets in a variety of flexible ways with minimal typing. Because R does partial argument matching, `mzrange`, `scanrange`, and `rtrange` can be specified in short form using `m=`, `s=`, and `t=`, respectively. If both a `scanrange` and `rtrange` are specified, then the `rtrange` specification takes precedence.

When specifying ranges, you may either enter a single number or a numeric vector. If a single number is entered, then the closest single scan or mass value is selected. If a vector is entered, then the range is set to the `range()` of the values entered. That allows specification of ranges using shortened, slightly non-standard syntax. For example, one could specify 400 to 500 seconds using any of the following: `t=c(400,500)`, `t=c(500,400)`, or `t=400:500`. Use of the sequence operator `:` can save several keystrokes when specifying ranges. However, while the sequence operator works well for specifying integer ranges, fractional ranges do not always work as well.

Value

A list with the following items:

- **mzrange**: numeric vector with start and end mass
- **masslab**: textual label of mass range
- **massidx**: integer vector of mass indecies
- **scanrange**: integer vector with start and end scans
- **scanlab**: textual label of scan range
- **scanidx**: integer vector of scan range
- **rtrange**: numeric vector of start and end times
- **timelab**: textual label of time range

Methods

```r
object = "xcmsRaw" profRange(object, mzrange = numeric(), rtrange = numeric(), scanrange = numeric(), ...
```
See Also

xcmsRaw-class

profStep-methods

Description

These methods get and set the m/z step for generating profile (matrix) data from raw mass spectral data. Smaller steps yield more precision at the cost of greater memory usage.

Methods

\[ \text{object = "xcmsRaw" profStep(object)} \]

See Also

xcmsRaw-class, profMethod

Examples

```r
## Not run:
library(faahKO)
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
xset <- xcmsRaw(cdffiles[1])

xset
plotSurf(xset, mass=c(200,500))

profStep(xset)<-0.1 ## decrease the bin size to get better resolution
plotSurf(xset, mass=c(200, 500))
## works nicer on high resolution data.
## End(Not run)
```

rawEIC-methods

Description

Generate extracted ion chromatogram for m/z values of interest. The raw data is used in contrast to getEIC which uses data from the profile matrix (i.e. values binned along the M/Z dimension).

Arguments

- object: xcmsRaw object
- mzrange: m/z range for EIC
- rtrange: retention time range for EIC
- scanrange: scan range for EIC
Value
A list of:
- scan: scan number
- intensity: added intensity values

Methods
- object = "xcmsRaw" rawEIC(object, mzrange = numeric(), rtrange = numeric(), scanrange = numeric())

Author(s)
Ralf Tautenhahn

See Also
- xcmsRaw-class

Description
Returns a matrix with columns for time, m/z, and intensity that represents the raw data from a chromatography mass spectrometry experiment.

Arguments
- object: The container of the raw data
- mzrange: Subset by m/z range
- rtrange: Subset by retention time range
- scanrange: Subset by scan index range
- log: Whether to log transform the intensities

Value
A numeric matrix with three columns: time, mz and intensity.

Methods
- object = "xcmsRaw" rawMat(object, mzrange = numeric(), rtrange = numeric(), scanrange = numeric(), log=FALSE)

Author(s)
Michael Lawrence

See Also
- plotRaw for plotting the raw intensities
retcor-methods

Correct retention time from different samples

Description

To correct differences between retention times between different samples, a number of methods exist in XCMS. retcor is the generic method.

Arguments

object    xcmsSet-class object
method    Method to use for retention time correction. See details.
...       Optional arguments to be passed along

Details

Different algorithms can be used by specifying them with the method argument. For example to use the approach described by Smith et al (2006) one would use: retcor(object, method="loess"). This is also the default.

Further arguments given by ... are passed through to the function implementing the method.

A character vector of nicknames for the algorithms available is returned by getOption("BioC")$xcms$retcor.methods If the nickname of a method is called "loess", the help page for that specific method can be accessed with ?retcor.loess.

Value

An xcmsSet object with corrected retention times.

Methods

object = "xcmsSet"    retcor(object, ...)

See Also

retcor.loess retcor.obiwarp xcmsSet-class,

retcor.obiwarp

Align retention times across samples with Obiwarp

Description

Calculate retention time deviations for each sample. It is based on the code at http://obi-warp.sourceforge.net/. However, this function is able to align multiple samples, by a center-star strategy.

For the original publication see

Chromatographic Alignment of ESI-LC-MS Proteomics Data Sets by Ordered Bijective Interpolated Warping John T. Prince and, Edward M. Marcotte Analytical Chemistry 2006 78 (17), 6140-6152
Arguments

- **object**: the `xcmsSet` object
- **plottype**: if deviation plot retention time deviation
- **profStep**: step size (in m/z) to use for profile generation from the raw data files
- **center**: the index of the sample all others will be aligned to. If center==NULL, the sample with the most peaks is chosen as default.
- **col**: vector of colors for plotting each sample
- **ty**: vector of line and point types for plotting each sample
- **response**: Responsiveness of warping. 0 will give a linear warp based on start and end points. 100 will use all bijective anchors
- **distFunc**: DistFunc function: cor (Pearson’s R) or cor_opt (default, calculate only 10% diagonal band of distance matrix, better runtime), cov (covariance), prd (product), euc (Euclidean distance)
- **gapInit**: Penalty for Gap opening, see below
- **gapExtend**: Penalty for Gap enlargement, see below
- **factorDiag**: Local weighting applied to diagonal moves in alignment.
- **factorGap**: Local weighting applied to gap moves in alignment.
- **localAlignment**: Local rather than global alignment
- **initPenalty**: Penalty for initiating alignment (for local alignment only) Default: 0

Default gap penalties: (gapInit, gapExtend) [by distFunc type]: ‘cor’ = ‘0.3,2.4’ ‘cov’ = ‘0,11.7’ ‘prd’ = ‘0.7.8’ ‘euc’ = ‘0.9,1.8’

Value

An `xcmsSet` object

Methods

```r
object = "xcmsSet" retcor(object, method="obiwarp", plottype = c("none", "deviation"), profStep=1, center=NULL, col = NULL, ty = NULL, response=1, distFunc="cor_opt", gapInit=NULL, gapExtend=NULL, factorDiag=2, factorGap=1, localAlignment=0, initPenalty=0)
```

See Also

- `xcmsSet-class`

Description

These two methods use “well behaved” peak groups to calculate retention time deviations for every time point of each sample. Use smoothed deviations to align retention times.
Arguments

object: the xcmsSet object
missing: number of missing samples to allow in retention time correction groups
extra: number of extra peaks to allow in retention time correction groups
smooth: either "loess" for non-linear alignment or "linear" for linear alignment
span: degree of smoothing for local polynomial regression fitting
family: if gaussian fitting is by least-squares with no outlier removal, and if symmetric
        a re-descending M estimator is used with Tukey's biweight function, allowing
        outlier removal
plottype: if deviation plot retention time deviation points and regression fit, and if mdevden
         also plot peak overall peak density and retention time correction peak density
col: vector of colors for plotting each sample
ty: vector of line and point types for plotting each sample

Value

An xcmsSet object

Methods

object = "xcmsSet" retcor(object, missing = 1, extra = 1, smooth = c("loess", "linear"), span = .2, family = c("gaussian", "symmetric"), plottype = c("none", "deviation", "mdevden"), col = NULL, ty = NULL)

See Also

xcmsSet-class, loess retcor.obiwarp

retexp

Set retention time window to a specified width

Description

Expands (or contracts) the retention time window in each row of a matrix as defined by the retmin
and retmax columns.

Usage

retexp(peakrange, width = 200)

Arguments

peakrange: matrix with columns retmin and retmax
width: new width for the window

Value

The altered matrix.

Author(s)

Colin A. Smith, <csmith@scripps.edu>
See Also

getEIC

Description

Return sample names for an object

Value

A character vector with sample names.

Methods

object = "xcmsEIC"  sampnames(object)
object = "xcmsSet"  sampnames(object)

See Also

xcmsSet-class, xcmsEIC-class

specDist-methods  Distance methods for xcmsSet, xcmsRaw and xsAnnotate

Description

There are several methods for calculating a distance between two sets of peaks in xcms. specDist is the generic method.

Arguments

object  a xcmsSet or xcmsRaw.
method  Method to use for distance calculation. See details.
...  mzabs, mzppm and parameters for the distance function.

Details

Different algorithms can be used by specifying them with the method argument. For example to use the "meanMZmatch" approach with xcmsSet one would use: specDist(object, peakIDs1, peakIDs2, method="meanMZmatch"). This is also the default.

Further arguments given by ... are passed through to the function implementing the method.

A character vector of nicknames for the algorithms available is returned by getOption("BioC")$xcms$specDist.methods. If the nickname of a method is called "meanMZmatch", the help page for that specific method can be accessed with ?specDist.meanMZmatch.
Value

mzabs  maximum absolute deviation for two matching peaks
mzppm  relative deviations in ppm for two matching peaks
symmetric  use symmetric pairwise m/z-matches only, or each match

Methods

object = "xcmsSet"  specDist(object, peakIDs1, peakIDs2,...)
object = "xsAnnotate"  specDist(object, PSpec1, PSpec2,...)

Author(s)

Joachim Kutzera, <jkutzer@ipb-halle.de>

Description

This method calculates the distance of two sets of peaks using the cosine-distance.

Usage

specDist.cosine(peakTable1, peakTable2, mzabs=0.001, mzppm=10, mzExp=0.6, intExp=3, nPdiff = 2, nPmin = 8, symmetric = FALSE)

Arguments

peakTable1  a Matrix containing at least m/z-values, row must be called "mz"
peakTable2  the matrix for the other m/z-values
mzabs  maximum absolute deviation for two matching peaks
mzppm  relative deviations in ppm for two matching peaks
symmetric  use symmetric pairwise m/z-matches only, or each match
mzExp  the exponent used for mz
intExp  the exponent used for intensity
nPdiff  the maximum nrow-difference of the two peak tables
nPmin  the minimum absolute sum of peaks from both peak tables

Details

The result is the cosine-distance of the product from weighted factors of \( m/z \) and intensity from matching peaks in the two peak tables. The factors are calculated as \( \text{wFact} = m/z^\text{mzExp} * \text{int}^\text{intExp} \). If no distance is calculated (for example because no matching peaks were found) the return-value is NA.

Methods

peakTable1 = "matrix", peakTable2 = "matrix"  specDist.cosine(peakTable1, peakTable2, mzabs = 0.001, mzppm = 10, mzExp = 0.6, intExp = 3, nPdiff = 2, nPmin = 8, symmetric = FALSE)
**specDist.meanMZmatch**

**Description**

This method calculates the distance of two sets of peaks.

**Usage**

```r
specDist.meanMZmatch(peakTable1, peakTable2, matchdist=1, matchrate=1, mzabs=0.001, mzppm=10, symmetric=TRUE)
```

**Arguments**

- `peakTable1`: a Matrix containing at least m/z-values, row must be called "mz"
- `peakTable2`: the matrix for the other m/z-values
- `mzabs`: maximum absolute deviation for two matching peaks
- `mzppm`: relative deviations in ppm for two matching peaks
- `symmetric`: use symmetric pairwise m/z-matches only, or each match
- `matchdist`: the weight for value one (see details)
- `matchrate`: the weight for value two

**Details**

The result of the calculation is a weighted sum of two values. Value one is the mean absolute difference of the matching peaks, value two is the relation of matching peaks and non matching peaks. If no distance is calculated (for example because no matching peaks were found) the return-value is NA.

**Methods**

```r
peakTable1 = "matrix", peakTable2 = "matrix" specDist.meanMZmatch(peakTable1, peakTable2, ...
```

**Author(s)**

Joachim Kutzera, <jkutzer@ipb-halle.de>
specDist.peakCount-methods

a Distance function based on matching peaks

Description

This method calculates the distance of two sets of peaks by just returning the number of matching peaks (m/z-values).

Usage

specDist.peakCount(peakTable1, peakTable2, mzabs=0.001, mzppm=10, symmetric=FALSE)

Arguments

peakTable1 a Matrix containing at least m/z-values, row must be called "mz"
peakTable2 the matrix for the other m/z-values
mzabs maximum absolute deviation for two matching peaks
mzppm relative deviations in ppm for two matching peaks
symmetric use symmetric pairwise m/z-matches only, or each match

Methods

peakTable1 = "matrix", peakTable2 = "matrix" specDist.peakCount(peakTable1, peakTable2, mzppm=10)

Author(s)

Joachim Kutzera, <jkutzer@ipb-halle.de>

specNoise

Calculate noise for a sparse continuum mass spectrum

Description

Given a sparse continuum mass spectrum, determine regions where no signal is present, substituting half of the minimum intensity for those regions. Calculate the noise level as the weighted mean of the regions with signal and the regions without signal. If there is only one raw peak, return zero.

Usage

specNoise(spec, gap = quantile(diff(spec[, "mz"]), 0.9))

Arguments

spec matrix with named columns mz and intensity
gap threshold above which to data points are considered to be separated by a blank region and not bridged by an interpolating line
specPeaks

Details

The default gap value is determined from the 90th percentile of the pair-wise differences between adjacent mass values.

Value

A numeric noise level

Author(s)

Colin A. Smith, <csmith@scripps.edu>

See Also

getSpec, specPeaks

Description

Given a spectrum, identify and list significant peaks as determined by several criteria.

Usage

specPeaks(spec, sn = 20, mzgap = 0.2)

Arguments

spec matrix with named columns mz and intensity
sn minimum signal to noise ratio
mzgap minimal distance between adjacent peaks, with smaller peaks being excluded

Details

Peaks must meet two criteria to be considered peaks: 1) Their s/n ratio must exceed a certain threshold. 2) They must not be within a given distance of any greater intensity peaks.

Value

A matrix with columns:

mz m/z at maximum peak intensity
intensity maximum intensity of the peak
fwhm full width at half max of the peak

Author(s)

Colin A. Smith, <csmith@scripps.edu>

See Also

getSpec, specNoise
split.xcmsRaw  

*Divide an xcmsRaw object*

**Description**

Divides the scans from a xcmsRaw object into a list of multiple objects. MS$n^k$ data is discarded.

**Arguments**

- **x**: xcmsRaw object
- **f**: factor such that factor(f) defines the scans which go into the new xcmsRaw objects
- **drop**: logical indicating if levels that do not occur should be dropped (if 'f' is a 'factor' or a list).
- **...**: further potential arguments passed to methods.

**Value**

A list of xcmsRaw objects.

**Methods**

```r
xr = "xcmsRaw" split(x, f, drop = TRUE, ...)
```

**Author(s)**

Steffen Neumann, <sneumann(at)ipb-halle.de>

**See Also**

- `xcmsRaw-class`

---

split.xcmsSet  

*Divide an xcmsSet object*

**Description**

Divides the samples and peaks from a xcmsSet object into a list of multiple objects. Group data is discarded.

**Arguments**

- **xs**: xcmsSet object
- **f**: factor such that factor(f) defines the grouping
- **drop**: logical indicating if levels that do not occur should be dropped (if 'f' is a 'factor' or a list).
- **...**: further potential arguments passed to methods.
**SSgauss**

**Value**
A list of `xcmsSet` objects.

**Methods**

```r
xs = "xcmsSet"         split(x, f, drop = TRUE, ...)
```

**Author(s)**
Colin A. Smith, <csmith@scripps.edu>

**See Also**
`xcmsSet-class`

---

### Description

This `selfStart` model evaluates the Gaussian model and its gradient. It has an initial attribute that will evaluate the initial estimates of the parameters \( \mu \), \( \sigma \), and \( h \).

**Usage**

```r
SSgauss(x, mu, sigma, h)
```

**Arguments**

- **x**: a numeric vector of values at which to evaluate the model
- **mu**: mean of the distribution function
- **sigma**: standard deviation of the distribution function
- **h**: height of the distribution function

**Details**

Initial values for \( \mu \) and \( h \) are chosen from the maximal value of \( x \). The initial value for \( \sigma \) is determined from the area under \( x \) divided by \( h \cdot \sqrt{2\pi} \).

**Value**

A numeric vector of the same length as \( x \). It is the value of the expression \( h \cdot \exp \left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \), which is a modified Gaussian function where the maximum height is treated as a separate parameter not dependent on \( \sigma \). If arguments \( \mu \), \( \sigma \), and \( h \) are names of objects, the gradient matrix with respect to these names is attached as an attribute named `gradient`.

**Author(s)**
Colin A. Smith, <csmith@scripps.edu>

**See Also**
`nls`, `selfStart`
stitch-methods

Correct gaps in data

Description

Fixes gaps in data due to calibration scans or lock mass. Automatically detects file type and calls the relevant method. The mzXML file keeps the data the same length in time but overwrites the lock mass scans. The netCDF version adds the scans back into the data thereby increasing the length of the data and correcting for the unseen gap.

Arguments

object: An xcmsRaw-class object
lockMass: A dataframe of locations of the gaps
freq: The intervals of the lock mass scans
start: The starting lock mass scan location, default is 1

Details

makeacqNum takes locates the gap using the starting lock mass scan and its intervals. This data frame is then used in stitch to correct for the gap caused by the lock mass. Correction works by using scans from either side of the gap to fill it in.

Value

stitch: A corrected xcmsRaw-class object
makeacqNum: A numeric vector of scan locations corresponding to lock Mass scans

Methods

object = "xcmsRaw" stitch(object, lockMass=numeric())

object = "xcmsRaw" makeacqNum(object, freq=numeric(), start=1)

Author(s)

Paul Benton, <hpaul.benton08@imperial.ac.uk>

Examples

## Not run: library(xcms)
library(faahKO) ## These files do not have this problem to correct for but just for an example
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
xr<-xcmsRaw(cdffiles[1])
xr

## Lets assume that the lockmass starts at 1 and is every 100 scans
lockMass<-xcms:::makeacqNum(xr, freq=100, start=1)

## these are equal
lockmass<-AutoLockMass(xr)
ob<-stitch(xr, lockMass)
ob
#plot the old data before correction
foo<-rawEIC(xr, m=c(200,210), scan=c(80,140))
plot(foo$scan, foo$intensity, type="h")

#plot the new corrected data to see what changed
foo<-rawEIC(ob, m=c(200,210), scan=c(80,140))
plot(foo$scan, foo$intensity, type="h")

## End(Not run)
Arguments

valCount Numeric vector representing the number of values per spectrum.

Value

An integer vector with the index (0-based) in the mz or intensity vectors indicating the start of a spectrum.

Author(s)

Johannes Rainer

verify.mzQuantML

Verify an mzQuantML file

Description

Export in XML data formats: verify the written data

Usage

verify.mzQuantML(filename, xsdfilename)

Arguments

filename filename (may include full path) for the output file. Pipes or URLs are not allowed.

xsdfilename Filename of the XSD to verify against (may include full path)

Details

The verify.mzQuantML() function will verify an PSI standard format mzQuantML document against the XSD schema, see http://www.psidev.info/mzquantml

Value

None.

See Also

write.mzQuantML
**write.cdf-methods**

**Save an xcmsRaw object to file**

**Description**

Write the raw data to a (simple) CDF file.

**Arguments**

- **object**: the xcmsRaw object
- **filename**: filename (may include full path) for the CDF file. Pipes or URLs are not allowed.

**Details**

Currently the only application known to read the resulting file is XCMS. Others, especially those which build on the AndiMS library, will refuse to load the output.

**Value**

None.

**Methods**

```r
object = "xcmsRaw" write.cdf(object, filename)
```

**See Also**

`xcmsRaw-class`, `xcmsRaw`.

---

**write.mzdata-methods**

**Save an xcmsRaw object to a file**

**Description**

Write the raw data to a (simple) mzData file.

**Arguments**

- **object**: the xcmsRaw object
- **filename**: filename (may include full path) for the mzData file. Pipes or URLs are not allowed.

**Details**

This function will export a given xcmsRaw object to an mzData file. The mzData file will contain a `<spectrumList>` containing the `<spectrum>` with mass and intensity values in 32 bit precision. Other formats are currently not supported. Any header information (e.g. additional `<software>` information or `<cvParams>`) will be lost. Currently, also any MSn information will not be stored.
Value

None.

Methods

\texttt{object = "xcmsRaw" write.mzdata(object, filename)}

See Also

\texttt{xcmsRaw-class, xcmsRaw}.

\texttt{write.mzQuantML-methods}

\emph{Save an \textit{xcmsSet} object to an PSI \textit{mzQuantML} file}

Description

Export in XML data formats: Write the processed data in an \textit{xcmsSet} to \textit{mzQuantML}.

Arguments

\texttt{object} \hspace{1cm} the \textit{xcmsRaw} or \textit{xcmsSet} object
\texttt{filename} \hspace{1cm} filename (may include full path) for the output file. Pipes or URLs are not allowed.

Details

The \texttt{write.mzQuantML()} function will write a (grouped) \textit{xcmsSet} into the PSI standard format \textit{mzQuantML}, see \url{http://www.psidev.info/mzquantml}

Value

None.

Methods

\texttt{object = "xcmsSet" write.mzQuantML(object, filename)}

See Also

\texttt{xcmsSet-class, xcmsSet, verify.mzQuantML}.
Description

Write the grouped xcmsSet to an mzTab file.

Arguments

object  the xcmsSet object
filename  filename (may include full path) for the mzTab file. Pipes or URLs are not allowed.

Details

The mzTab file format for MS-based metabolomics (and proteomics) is a lightweight supplement to the existing standard XML-based file formats (mzML, mzIdentML, mzQuantML), providing a comprehensive summary, similar in concept to the supplemental material of a scientific publication. mzTab files from xcms contain small molecule sections together with experimental metadata and basic quantitative information. The format is intended to store a simple summary of the final results.

Value

None.

Usage

object = "xcmsSet"  writeMzTab(object, filename)

See Also

xcmsSet-class, xcmsSet.

Examples

library(faahKO)
xs <- group(faahko)
mzt <- data.frame(character(0))
mzt <- xcms:::mzTabHeader(mzt,
    version="1.1.0", mode="Complete", type="Quantification",
    description="faahKO",
    xset=xs)
mzt <- xcms:::mzTabAddSME(mzt, xs)
xcms:::writeMzTab(mzt, "faahKO.mzTab")
xcms-deprecated  Deprecated functions in package ‘xcms’

Description

These functions are provided for compatibility with older versions of ‘xcms’ only, and will be defunct at the next release.

Details

The following functions/methods are deprecated.

- xcmsPapply: this function is no longer available and the use of bplapply is suggested.

xcmsEIC-class  Class xcmsEIC, a class for multi-sample extracted ion chromatograms

Description

This class is used to store and plot parallel extracted ion chromatograms from multiple sample files. It integrates with the xcmsSet class to display peak area integrated during peak identification or fill-in.

Objects from the Class

Objects can be created with the getEIC method of the xcmsSet class. Objects can also be created by calls of the form new("xcmsEIC", ...).

Slots

eic: list containing named entries for every sample. for each entry, a list of two column EIC matrices with retention time and intensity
mzrange: two column matrix containing starting and ending m/z for each EIC
rtrange: two column matrix containing starting and ending time for each EIC
rt: either "raw" or "corrected" to specify retention times contained in the object
groupnames: group names from xcmsSet object used to generate EICs

Methods

groupnames  signature(object = "xcmsEIC"): get groupnames slot
mzrange  signature(object = "xcmsEIC"): get mzrange slot
plot  signature(x = "xcmsEIC"): plot the extracted ion chromatograms
rtrange  signature(object = "xcmsEIC"): get rtrange slot
sampnames  signature(object = "xcmsEIC"): get sample names

Note

No notes yet.
Description

Data sources which read data from a file should inherit from this class. The xcms package provides classes to read from netCDF, mzData, mzXML, and mzML files using xcmsFileSource.

This class should be considered virtual and will not work if passed to loadRaw-methods. The reason it is not explicitly virtual is that there does not appear to be a way for a class to be both virtual and have a data part (which lets functions treat objects as if they were character strings).

This class validates that a file exists at the path given.

Objects from the Class

xcmsFileSource objects should not be instantiated directly. Instead, create subclasses and instantiate those.

Slots

.Data: Object of class "character". File path of a file from which to read raw data as the object’s data part

Extends

Class "character", from data part. Class "xcmsSource", directly.

Methods

xcmsSource signature(object = "character"): Create an xcmsFileSource object referencing the given file name.

Author(s)

Daniel Hackney <dan@haxney.org>

See Also

xcmsSource
**xcmsFragments**

Construct for xcmsFragments objects which holds Tandem MS peaks

---

**Description**

EXPERIMENTAL FEATURE

xcmsFragments is an object similar to xcmsSet, which holds peaks picked (or collected) from one or several xcmsRaw objects.

There are still discussions going on about the exact API for MS^n data, so this is likely to change in the future. The code is not yet pipeline-ified.

**Usage**

```r
xcmsFragments(xs, ...)```

**Arguments**

- `xs` A `xcmsSet-class` object which contains picked ms1-peaks from one or several experiments
- `...` further arguments to the `collect` method

**Details**

After running `collect(xFragments,xSet)` the peaktable of the xcmsFragments includes the ms1Peaks from all experiments stored in a xcmsSet-object. Further it contains the relevant MSn-peaks from the xcmsRaw-objects, which were created temporarily with the paths in xcmsSet.

**Value**

An xcmsFragments object.

**Author(s)**

Joachim Kutzera, Steffen Neumann, <sneumann@ipb-halle.de>

**See Also**

`xcmsFragments-class`, `collect`
Class xcmsFragments, a class for handling Tandem MS and MS^n data

Description

This class is similar to xcmsSet because it stores peaks from a number of individual files. However, xcmsFragments keeps Tandem MS and e.g. Ion Trap or Orbitrap MS^n peaks, including the parent ion relationships.

Objects from the Class

Objects can be created with the xcmsFragments constructor and filled with peaks using the collect method.

Slots

peaks: matrix with columns peakID (MS1 parent in corresponding xcmsSet), MSnParentPeakID (parent peak within this xcmsFragments), msLevel (e.g. 2 for Tandem MS), rt (retention time in case of LC data), mz (fragment mass-to-charge), intensity (peak intensity extracted from the original xcmsSet), sample (the index of the rawData-file).

MS2spec: This is a list of matrices. Each matrix in the list is a single collected spectra from collect. The column ID’s are mz, intensity, and full width half maximum(fwhm). The fwhm column is only relevant if the spectra came from profile data.

specinfo: This is a matrix with reference data for the spectra in MS2spec. The column id’s are preMZ, AccMZ, rtmin, rtmax, ref, CollisionEnergy. The preMZ is precursor mass from the MS1 scan. This mass is given by the XML file. With some instruments this mass is only given as nominal mass, therefore a AccMZ is given which is a weighted average mass from the MS1 scan of the collected spectra. The retention time is given by rtmin and rtmax. The ref column is a pointer to the MS2spec matrix spectra. The collisionEnergy column is the collision Energy for the spectra.

Methods

collect signature(object = "xcmsFragments"): gets a xcmsSet-object, collects ms1-peaks from it and the msn-peaks from the corresponding xcmsRaw-files.

plotTree signature(object = "xcmsFragments"): prints a (text based) pseudo-tree of the peak-table to display the dependencies of the peaks among each other.

show signature(object = "xcmsFragments"): print a human-readable description of this object to the console.

Note

No notes yet.

Author(s)

S. Neumann, J. Kutzera
References

A parallel effort in metabolite profiling data sharing: http://metlin.scripps.edu/

See Also

xcmsRaw

xcmsPapply Deprecated: xcmsPapply

Description

This function is deprecated, use bplapply instead.

An apply-like function which uses Rmpi to distribute the processing evenly across a cluster. Will use a non-MPI version if distributed processing is not available.

Usage

xcmsPapply(arg_sets, papply_action, papply_commondata = list(),
    show_errors = TRUE, do_trace = FALSE, also_trace = c())

Arguments

arg_sets a list, where each item will be given as an argument to papply\_action
papply_action A function which takes one argument. It will be called on each element of arg\sets
papply_commondata A list containing the names and values of variables to be accessible to the papply\_action. 'attach' is used locally to import this list.
show_errors If set to TRUE, overrides Rmpi's default, and messages for errors which occur in R slaves are produced.
do_trace If set to TRUE, causes the papply\_action function to be traced. i.e. Each statement is output before it is executed by the slaves.
also_trace If supplied an array of function names, as strings, tracing will also occur for the specified functions.

Details

Similar to apply and lapply, applies a function to all items of a list, and returns a list with the corresponding results.

Uses Rmpi to implement a pull idiom in order to distribute the processing evenly across a cluster. If Rmpi is not available, or there are no slaves, implements this as a non-parallel algorithm.

xcmsPapply is a modified version of the papply function from package papply 0.2 (Duane Currie). Parts of the slave function were wrapped in try() to make it failsafe and progress output was added.

Make sure Rmpi was installed properly by executing the example below. Rmpi was tested with

- OpenMPI: Unix, http://www.open-mpi.org/, don't forget to export MPI_ROOT before installing Rmpi e.g. export MPI_ROOT=/usr/lib/openmpi
### Value

A list of return values from `papply\_action`. Each value corresponds to the element of `arg\_sets` used as a parameter to `papply\_action`.

### Note

Does not support distributing recursive calls in parallel. If `papply` is used inside `papply\_action`, it will call a non-parallel version.

### Author(s)

Duane Currie <duane.currie@acadiau.ca>, modified by Ralf Tautenhahn <rtautenh@ipb-halle.de>.

### References


### Examples

```r
## Not run:
library(Rmpi)
library(xcms)

number_lists <- list(1:10, 4:40, 2:27)

mpi.spawn.Rslaves(nslaves=2)

results <- xcmsPapply(number_lists, sum)

results

mpi.close.Rslaves()

## End(Not run)
```

---

### xcmsPeaks-class

A matrix of peaks

#### Description

A matrix of peak information. The actual columns depend on how it is generated (i.e. the `findPeaks` method).

#### Objects from the Class

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

#### Slots

- `.Data`: The matrix holding the peak information
**Extend**


**Methods**

None yet. Some utilities for working with peak data would be nice.

**Author(s)**

Michael Lawrence

**See Also**

*findPeaks* for detecting peaks in an *xcmsRaw*.

---

**xcmsRaw**  
*Constructor for xcmsRaw objects which reads NetCDF/mzXML files*

**Description**

This function handles the task of reading a NetCDF/mzXML file containing LC/MS or GC/MS data into a new *xcmsRaw* object. It also transforms the data into profile (matrix) mode for efficient plotting and data exploration.

**Usage**

```r
xcmsRaw(filename, profstep = 1, profmethod = "bin", profparam = list(), includeMSn=FALSE, mslevel=NULL, scanrange=NULL)

depCopy(object)
```

**Arguments**

- `filename`  
  path name of the NetCDF or mzXML file to read
- `profstep`  
  step size (in m/z) to use for profile generation
- `profmethod`  
  method to use for profile generation
- `profparam`  
  extra parameters to use for profile generation
- `includeMSn`  
  only for XML file formats: also read MS^n$ (Tandem-MS of Ion-/Orbi- Trap spectra)
- `mslevel`  
  move data from mslevel into normal MS1 slots, e.g. for peak picking and visualisation
- `scanrange`  
  scan range to read
- `object`  
  An *xcmsRaw* object
Details

The scanrange to import can be restricted, otherwise all MS1 data is read. If profstep is set to 0, no profile matrix is generated. Unless includeMSn=TRUE only first level MS data is read, not MS/MS, etc.

deeperCopy(xraw) will create a copy of the xcmsRaw object with its own copy of mz and intensity data in xraw$env.

Value

A xcmsRaw object.

Author(s)

Colin A. Smith, <csmith@scripps.edu>

References


mzXML file format: http://sashimi.sourceforge.net/software_glossolalia.html

PSI-MS working group who developed mzData and mzML file formats: http://www.psidev.info/index.php?q=node/80


See Also

xcmsRaw-class, profStep, profMethod xcmsFragments

Examples

## Not run:
library(xcms)
library(faahKO)
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
xr<-xcmsRaw(cdffiles[1])
xr
##This gives some information about the file
names(attributes(xr))
## Lets have a look at the structure of the object
str(xr)
##same but with a preview of each slot in the object
##SO... lets have a look at how this works
head(xr$scanindex)
# [1] 0 429 860 1291 1718 2140
xr$env$mz[425:430]
# [1] 596.3 597.0 597.3 598.1 599.3 200.1
##We can see that the 429 index is the last mz of scan 1 therefore...

mz.scan1<-xr$env$mz[(1+xr$scanindex[1]):xr$scanindex[2]]
intensity.scan1<-xr$env$intensity[(1+xr$scanindex[1]):xr$scanindex[2]]
plot(mz.scan1, intensity.scan1, type="h", main=paste("Scan 1 of file", basename(cdffiles[1]), sep=""))
## The xcmsRaw-class

This class handles processing and visualization of the raw data from a single LC/MS or GS/MS run. It includes methods for producing a standard suite of plots including individual spectra, multi-scan average spectra, TIC, and EIC. It will also produce a feature list of significant peaks using matched filtration.

### Objects from the Class

Objects can be created with the `xcmsRaw` constructor which reads data from a NetCDF file into a new object.

### Slots

- **acquisitionNum**: acquisitionNum
- **env**: environment with three variables: `mz` - concatenated m/z values for all scans, `intensity` - corresponding signal intensity for each m/z value, and `profile` - matrix representation of the intensity values with columns representing scans and rows representing equally spaced m/z values
- **filepath**: Path to the raw data file
- **gradient**: matrix with first row, `time`, containing the time point for interpolation and successive columns representing solvent fractions at each point
- **msnAcquisitionNum**: for each scan a unique acquisition number as reported via "spectrum id" (mzData) or "<scan num=...>" and "<scanOrigin num=...>") (mzXML)
- **msnCollisionEnergy**: "CollisionEnergy" (mzData) or "collisionEnergy" (mzXML)
- **msnLevel**: for each scan the “msLevel” (both mzData and mzXML)
- **msnPrecursorCharge**: "ChargeState" (mzData) and "precursorCharge" (mzXML)
- **msnPrecursorIntensity**: "Intensity" (mzData) or "precursorIntensity" (mzXML)
- **msnPrecursorMz**: "MassToChargeRatio" (mzData) or "precursorMz" (mzXML)
- **msnPrecursorScan**: "spectrumRef" (both mzData and mzXML)
- **msnRt**: Retention time of the scan
- **msnScanindex**: msnScanindex
- **mzrange**: numeric vector of length 2 with minimum and maximum m/z values represented in the profile matrix
- **polarity**: polarity
- **profmethod**: character value with name of method used for generating the profile matrix
**Methods**

**findPeaks** signature(object = "xcmsRaw"): feature detection using matched filtration in the chromatographic time domain

**getEIC** signature(object = "xcmsRaw"): get extracted ion chromatograms in specified m/z ranges. This will return the total ion chromatogram (TIC) if the m/z range corresponds to the full m/z range (i.e. sum of all signals per retention time across all m/z).

**getPeaks** signature(object = "xcmsRaw"): get data for peaks in specified m/z and time ranges

**getScan** signature(object = "xcmsRaw"): get m/z and intensity values for a single mass scan

**getSpec** signature(object = "xcmsRaw"): get average m/z and intensity values for multiple mass scans

**image** signature(x = "xcmsRaw"): get data for peaks in specified m/z and time ranges

**levelplot** Create an image of the raw (profile) data m/z against retention time, with the intensity color coded.

**mslevel** Getter method for the mslevel slot.

**plotChrom** signature(object = "xcmsRaw"): plot a chromatogram from profile data

**plotRaw** signature(object = "xcmsRaw"): plot locations of raw intensity data points

**plotScan** signature(object = "xcmsRaw"): plot a mass spectrum of an individual scan from the raw data

**plotSpec** signature(object = "xcmsRaw"): plot a mass spectrum from profile data

**plotSurf** signature(object = "xcmsRaw"): experimental method for plotting 3D surface of profile data with rgl.

**plotTIC** signature(object = "xcmsRaw"): plot total ion count chromatogram

**profinfo** signature(object = "xcmsRaw"): returns a list containing the profile generation method and step (profile m/z step size) and eventual additional parameters to the profile function.

**profMedFilt** signature(object = "xcmsRaw"): median filter profile data in time and m/z dimensions

**profMethod<-** signature(object = "xcmsRaw"): change the method of generating the profile matrix

**profMethod** signature(object = "xcmsRaw"): get the method of generating the profile matrix

**profMz** signature(object = "xcmsRaw"): get vector of m/z values for each row of the profile matrix

**profRange** signature(object = "xcmsRaw"): interpret flexible ways of specifying subsets of the profile matrix
The `xcmsSet` function handles the construction of `xcmsSet` objects. It finds peaks in batch mode and pre-sorts files from subdirectories into different classes suitable for grouping.

### Usage

```r
cmsSet(files = NULL, snames = NULL, sclass = NULL, phenoData = NULL, 
profmethod = "bin", profparam = list(), 
polarity = NULL, lockMassFreq=FALSE, 
mslevel=NULL, nSlaves=0, progressCallback=NULL, 
scanrange = NULL, BPPARAM = bpparam(), ...)
```

### Arguments

- **files**: path names of the NetCDF/mzXML files to read
- **snames**: sample names. By default the file name without extension is used.
- **sclass**: sample classes.

### Description

This function handles the construction of `xcmsSet` objects. It finds peaks in batch mode and pre-sorts files from subdirectories into different classes suitable for grouping.

### Note

No notes yet.

### Author(s)

Colin A. Smith, <csmith@scripps.edu>, Johannes Rainer <johannes.rainer@eurac.edu>

### References

A parallel effort in metabolite profiling data sharing: [http://metlin.scripps.edu/](http://metlin.scripps.edu/)

### See Also

`xcmsRaw`
The default values of the files, snames, sclass, and phenoData arguments cause the function to recursively search for readable files. The filename without extension is used for the sample name. The subdirectory path is used for the sample class. If the files contain both positive and negative spectra, the polarity can be selected explicitly. The default (NULL) is to read all scans.

If phenoData is provided, it is stored to the phenoData slot of the returned xcmsSet class. If that data.frame contains a column named “class”, its content will be returned by the sampclass method and thus be used for the group/class assignment of the individual files (e.g. for peak grouping etc.). For more details see the help of the xcmsSet-class.

The step size (in m/z) to use for profile generation can be submitted either using the profparam argument (e.g. profparam=list(step=0.1)) or by submitting step=0.1. By specifying a value of 0 the profile matrix generation can be skipped.

The feature/peak detection algorithm can be specified with the method argument which defaults to the “matchFilter” method (findPeaks.matchedFilter). Possible values are returned bygetOption("BioC")$xcms$findPeaks.methods.

The lock mass correction allows for the lock mass scan to be added back in with the last working scan. This correction gives better reproducibility between sample sets.

Value

A xcmsSet object.

Note

The arguments profmethod and profparam have no influence on the feature/peak detection. The step size parameter step for the profile generation in the findPeaks.matchedFilter peak detection algorithm can be passed using the ....
Class xcmsSet, a class for preprocessing peak data

Description
This class transforms a set of peaks from multiple LC/MS or GC/MS samples into a matrix of preprocessed data. It groups the peaks and does nonlinear retention time correction without internal standards. It fills in missing peak values from raw data. Lastly, it generates extracted ion chromatograms for ions of interest.

Details
The phenoData slot (and phenoData parameter in the xcmsSet function) is intended to contain a data.frame describing all experimental factors, i.e. the samples along with their properties. If this data.frame contains a column named “class”, this will be returned by the sampclass method and will thus be used by all methods to determine the sample grouping/class assignment (e.g. to define the colors in various plots or for the group method).

The sampclass<- method adds or replaces the “class” column in the phenoData slot. If a data.frame is submitted to this method, the interaction of its columns will be stored into the “class” column.

Also, similar to other classes in Bioconductor, the $ method can be used to directly access all columns in the phenoData slot (e.g. use xset$name on a xcmsSet object called “xset” to extract the values from a column named “name” in the phenoData slot).

Objects from the Class
Objects can be created with the xcmsSet constructor which gathers peaks from a set NetCDF files. Objects can also be created by calls of the form new("xCmsSet", ...).

Slots
peaks: matrix containing peak data
filled: a vector with peak indices of peaks which have been added by a fillPeaks method,
groups: matrix containing statistics about peak groups
groupidx: list containing indices of peaks in each group
phenoData: a data frame containing the experimental design factors
rt: list containing two lists, raw and corrected, each containing retention times for every scan of every sample
filepaths: character vector with absolute path name of each NetCDF file
profinfo: list containing the values method - profile generation method, and step - profile m/z step size and eventual additional parameters to the profile function.
dataCorrection logical vector filled if the waters Lock mass correction parameter is used.
polarity: a string ("positive" or "negative" or NULL) describing whether only positive or negative scans have been used reading the raw data.

progressInfo: progress informations for some xcms functions (for GUI)
progressCallback: function to be called, when progressInfo changes (for GUI)
mslevel: Numeric representing the MS level on which the peak picking was performed (by default on MS1). This slot should be accessed through its getter method mslevel.

scanrange: Numeric of length 2 specifying the scan range (or NULL for the full range). This slot should be accessed through its getter method scanrange.

Methods

c signature("xcmsSet"): combine objects together
filepaths<- signature(object = "xcmsSet"): set filepaths slot
filepaths signature(object = "xcmsSet"): get filepaths slot
diffreport signature(object = "xcmsSet"): create report of differentially regulated ions including EICs
fillPeaks signature(object = "xcmsSet"): fill in peak data for groups with missing peaks
getEIC signature(object = "xcmsSet"): get list of EICs for each sample in the set
getXcmsRaw signature(object = "xcmsSet", sampleidx = 1,profmethod = profMethod(object), profstep = profStep(object), profparam = profinfo(object), mslevel = NULL, scanrange = NULL,rt = c("corrected", "raw"), BPPARAM = bpparam()): read the raw data for one or more files in the xcmsSet and return it. The default parameters will apply all settings used in the original xcmsSet call to generate the xcmsSet object to be applied also to the raw data. Parameter sampleidx allows to specify which raw file(s) should be loaded. Argument BPPARAM allows to setup parallel processing.
groupidx<- signature(object = "xcmsSet"): set groupidx slot
groupidx signature(object = "xcmsSet"): get groupidx slot
groupnames signature(object = "xcmsSet"): get textual names for peak groups
groups<- signature(object = "xcmsSet"): set groups slot
groups signature(object = "xcmsSet"): get groups slot
groupval signature(object = "xcmsSet"): get matrix of values from peak data with a row for each peak group
group signature(object = "xcmsSet"): find groups of peaks across samples that share similar m/z and retention times

mslevel Getter method for the mslevel slot.
peaks<- signature(object = "xcmsSet"): set peaks slot
peaks signature(object = "xcmsSet"): get peaks slot
plotrt signature(object = "xcmsSet"): plot retention time deviation profiles
profinfo<- signature(object = "xcmsSet"): set profinfo slot
profinfo signature(object = "xcmsSet"): get profinfo slot
profMethod signature(object = "xcmsSet"): extract the method used to generate the profile matrix.
profStep signature(object = "xcmsSet"): extract the profile step used for the generation of the profile matrix.
retcor signature(object = "xcmsSet"): use initial grouping of peaks to do nonlinear loess retention time correction
**sampclass**<- signature(object = "xcmsSet"): Replaces the column “class” in the phenoData slot. See details for more information.

**sampclass** signature(object = "xcmsSet"): Returns the content of the column “class” from the phenoData slot or, if not present, the interaction of the experimental design factors (i.e. of the phenoData data.frame). See details for more information.

**phenoData**<- signature(object = "xcmsSet"): set the phenoData slot

**phenoData** signature(object = "xcmsSet"): get the phenoData slot

**progressCallback**<- signature(object = "xcmsSet"): set the progressCallback slot

**progressCallback** signature(object = "xcmsSet"): get the progressCallback slot

**scanrange** Getter method for the scanrange slot.

**sampnames**<- signature(object = "xcmsSet"): set rownames in the phenoData slot

**sampnames** signature(object = "xcmsSet"): get rownames in the phenoData slot

**split** signature("xcmsSet"): divide the xcmsSet into a list of xcmsSet objects depending on the provided factor. Note that only peak data will be preserved, i.e. eventual peak grouping information will be lost.

**object$name, object$name<-value** Access and set name column in phenoData

**object[, i]** Conducts subsetting of a xcmsSet instance. Only subsetting on columns, i.e. samples, is supported. Subsetting is performed on all slots, also on groups and groupidx. Parameter i can be an integer vector, a logical vector or a character vector of sample names (matching sampnames).

**Note**

No notes yet.

**Author(s)**

Colin A. Smith, <csmith@scripps.edu>, Johannes Rainer <johannes.rainer@eurac.edu>

**References**

A parallel effort in metabolite profiling data sharing: [http://metlin.scripps.edu/](http://metlin.scripps.edu/)

**See Also**

xcmsSet

---

**xcmsSource-class** Virtual class for raw data sources

**Description**

This virtual class provides an implementation-independent way to load mass spectrometer data from various sources for use in an xcmsRaw object. Subclasses can be defined to enable data to be loaded from user-specified sources. The virtual class xcmsFileSource is included out of the box which contains a file name as a character string.

When implementing child classes of xcmsSource, a corresponding loadRaw-methods method must be provided which accepts the xcmsSource child class and returns a list in the format described in loadRaw-methods.
**Objects from the Class**

A virtual Class: No objects may be created from it.

**Author(s)**

Daniel Hackney, <dan@haxney.org>

**See Also**

`xcmsSource-methods` for creating `xcmsSource` objects in various ways.

---

**xcmsSource-methods**  *Create an `xcmsSource` object in a flexible way*

**Description**

Users can define alternate means of reading data for `xcmsRaw` objects by creating new implementations of this method.

**Methods**

`signature(object = "xcmsSource")` Pass the object through unmodified.

**Author(s)**

Daniel Hackney, <dan@haxney.org>

**See Also**

`xcmsSource`
Index

*Topic classes
  xcmsEIC-class, 72
  xcmsFileSource-class, 73
  xcmsFragments-class, 75
  xcmsPeaks-class, 77
  xcmsRaw-class, 80
  xcmsSet-class, 84
  xcmsSource-class, 86

*Topic file
  calibrate-methods, 6
  diffreport-methods, 8
  fillPeaks-methods, 11
  fillPeaks.chrom-methods, 11
  fillPeaks.MSW-methods, 12
  getEIC-methods, 29
  getXcmsRaw-methods, 32
  group.density, 34
  group.mzClust, 34
  group.nearest, 35
  groupnames-methods, 37
  peakTable-methods, 42
  retcor.peakgroups-methods, 57
  sampnames-methods, 59
  verify.mzQuantM, 68
  write.cdf-methods, 69
  write.mzdata-methods, 69
  write.mzQuantML-methods, 70
  writeMzTab, 71
  xcmsFileSource-class, 73
  xcmsFragments, 74
  xcmsRaw, 78
  xcmsSet, 82

*Topic iplot
  image-methods, 38
  levelPlot-methods, 39
  plot.xcmsEIC, 44
  plotChrom-methods, 45
  plotPeaks-methods, 46
  plotRaw-methods, 48
  plotrt-methods, 48
  plotScan-methods, 49
  plotSpec-methods, 50
  plotSurf-methods, 50
  plotTIC-methods, 51

*Topic lockmass
  AutoLockMass-methods, 4

*Topic manip
  AutoLockMass-methods, 4
  c-methods, 5
  getPeaks-methods, 30
  getScan-methods, 31
  getSpec-methods, 32
  groupval-methods, 37
  medianFilter, 40
  msn2xcmsRaw, 41
  profMedFilt-methods, 52
  profMethod-methods, 52
  profRange-methods, 53
  profStep-methods, 54
  retexp, 58
  specNoise, 62
  specPeaks, 63
  split.xcmsRaw, 64
  split.xcmsSet, 64
  stitch-methods, 66

*Topic methods
  absent-methods, 4
  AutoLockMass-methods, 4
  calibrate-methods, 6
  collect-methods, 7
  diffreport-methods, 8
  fillPeaks-methods, 11
  fillPeaks.chrom-methods, 11
  fillPeaks.MSW-methods, 12
  findMZ, 13
  findneutral, 14
  findPeaks-methods, 16
  findPeaks.addPredictedIsotopeFeatures-methods, 17
  findPeaks.centWave-methods, 19
  findPeaks.centWaveWithPredictedIsotopeROIs-methods, 20

88
findPeaks.massifquant-methods, 23
findPeaks.matchedFilter-methods, 26
findPeaks.MS1-methods, 27
findPeaks.MSW-methods, 28
getEIC-methods, 29
getPeaks-methods, 30
group-methods, 33
group.density, 34
group.mzClust, 34
group.nearest, 35
groupnames-methods, 37
groupval-methods, 37
loadRaw-methods, 39
peakPlots-methods, 42
peakTable-methods, 42
plot.xcmsEIC, 44
plotChrom-methods, 45
plotEIC-methods, 45
plotPeaks-methods, 46
plotRaw-methods, 48
plotrt-methods, 48
plotScan-methods, 49
plotSpec-methods, 50
plotSurf-methods, 50
plotTIC-methods, 51
profMedFilt-methods, 52
profMethod-methods, 52
profRange-methods, 53
profStep-methods, 54
rawEIC-methods, 54
rawMat-methods, 55
retcor-methods, 56
retcor.obiwarp, 56
retcor.peakgroups-methods, 57
sampnames-methods, 59
specDist-methods, 59
specDist.cosine, 60
specDist.meanMZmatch, 61
specDist.peakCount-methods, 62
stitch-methods, 66
write.cdf-methods, 69
write.mzdata-methods, 69
write.mzQuantML-methods, 70
xcmsSource-methods, 87

*Topic **models**
etg, 10

*Topic **nonlinear**
SSgauss, 65
[, xcmsSet, ANY, ANY, ANY-method
(xcmsSet-class), 84
[, xcmsSet-method (xcmsSet-class), 84
$, xcmsSet-method (xcmsSet-class), 84
$<-, xcmsSet-method (xcmsSet-class), 84
absent (absent-methods), 4
absent, xcmsSet-method (absent-methods), 4
absent-methods, 4
array, 78
AutoLockMass (AutoLockMass-methods), 4
AutoLockMass, xcmsRaw-method
(AutoLockMass-methods), 4
AutoLockMass-methods, 4
bplapply, 72, 76
c, 85
c, c-methods (c-methods), 5
c-methods, 5
c.xcmsSet (c-methods), 5
calibrate (calibrate-methods), 6
calibrate, xcmsSet-method
(calibrate-methods), 6
calibrate-methods, 6
character, 73
collect, 74, 75
collect (collect-methods), 7
collect, xcmsFragments-method
(collect-methods), 7
collect, xcmsRaw-method
(collect-methods), 7
collect-methods, 7
deepCopy (xcmsRaw), 78
deepCopy, xcmsRaw-method (xcmsRaw), 78
density, 34
diffreport, 4, 43, 85
diffreport (diffreport-methods), 8
diffreport, xcmsSet-method
(diffreport-methods), 8
diffreport-methods, 8
etg, 10
filepaths (xcmsSet-class), 84
filepaths, xcmsSet-method
(xcmsSet-class), 84
filepaths<- (xcmsSet-class), 84
filepaths<-, xcmsSet-method
(xcmsSet-class), 84
fillPeaks, 4, 12, 13, 84, 85
INDEX

getXcmsRaw, xcmsSet-method
  (getXcmsRaw-methods), 32
getXcmsRaw-methods, 32
group, 4, 84, 85
group (group-methods), 33
group, xcmsSet-method (group-methods), 33
group-methods, 33
group.density, 33, 34, 36
group.density, xcmsSet-method
  (group.density), 34
group.mzClust, 33, 34, 36
group.mzClust, xcmsSet-method
  (group.mzClust), 34
group.nearest, 33, 35
group.nearest, xcmsSet-method
  (group.nearest), 35
groupidx (xcmsSet-class), 84
groupidx, xcmsSet-method
  (xcmsSet-class), 84
groupidx<-, xcmsSet-method
  (xcmsSet-class), 84
groupnames, 72, 85
groupnames (groupnames-methods), 37
groupnames, xcmsEIC-method
  (groupnames-methods), 37
groupnames, xcmsSet-method
  (groupnames-methods), 37
groupnames-methods, 37
groups (xcmsSet-class), 84
groups, xcmsSet-method (xcmsSet-class), 84
groups<-, xcmsSet-method (xcmsSet-class), 84
groupval, 42, 85
groupval (groupval-methods), 37
groupval, xcmsSet-method
  (groupval-methods), 37
groupval-methods, 37
image, 81
image, xcmsRaw-method (image-methods), 38
image-methods, 38
levelplot, 81
levelplot (xcmsRaw-class), 80
levelplot, xcmsRaw-method
  (levelplot-methods), 39
levelplot, xcmsSet-method
  (levelplot-methods), 39
levelplot-methods, 39
loadRaw (loadRaw-methods), 39
loadRaw, xcmsSource-method
  (loadRaw-methods), 39
loadRaw-methods, 39
loess, 58
makeacqNum (stitch-methods), 66
makeacqNum, xcmsRaw-method
  (stitch-methods), 66
matrix, 78
medianFilter, 40, 52
mslevel (xcmsSet-class), 84
mslevel, xcmsRaw-method (xcmsRaw-class), 80
mslevel, xcmsSet-method (xcmsSet-class), 84
msn2xcmsRaw, 41
mt.teststat, 8, 9
MulticoreParam, 83
mzrange (xcmsEIC-class), 72
mzrange, xcmsEIC-method (xcmsEIC-class), 72
nls, 65
palette, 9
pdf, 44
peakDetectionCWT, 29
peakPlots, xcmsSet-method
  (peakPlots-methods), 42
peakPlots-methods, 42
peaks (xcmsSet-class), 84
peaks, xcmsSet-method (xcmsSet-class), 84
peaks<-, xcmsSet-method (xcmsSet-class), 84
peakTable (peakTable-methods), 42
peakTable, xcmsSet-method
  (peakTable-methods), 42
peakTable-methods, 42
phenoData (xcmsSet-class), 84
phenoData, xcmsSet-method
  (xcmsSet-class), 84
phenoData<-, xcmsSet-class (ANY-method)
  (xcmsSet-class), 84
phenoData<-, xcmsSet-method
  (xcmsSet-class), 84
plot, 72
plot, plot-methods (plot.xcmsEIC), 44
plot.xcmsEIC, 44
plotChrom, 45, 52, 81
plotChrom (plotChrom-methods), 45
plotChrom, xcmsRaw-method
(plotChrom-methods), 45
plotChrom-methods, 45
plotEIC, xcmsRaw-method
(plotEIC-methods), 45
plotEIC-methods, 45
plotPeaks, xcmsRaw-method
(plotPeaks-methods), 46
plotPeaks-methods, 46
plotRaw, xcmsRaw-method
(plotRaw-methods), 48
plotRaw-methods, 48
plotrt
(plotrt-methods), 48
plotrt-methods, 48
plotScan, xcmsRaw-method
(plotScan-methods), 49
plotScan-methods, 49
plotSpec, xcmsRaw-method
(plotSpec-methods), 50
plotSpec-methods, 50
plotSurf, xcmsRaw-method
(plotSurf-methods), 50
plotSurf-methods, 50
plotTIC
(plotTIC-methods), 51
plotTIC-methods, 51
plotTree, xcmsFragments-class
(plotTree-methods), 75
plotTree, xcmsFragments-method
(xcmsFragments-class), 75
png, 44
postscript, 44
present (absent-methods), 4
present, xcmsSet-method
(absent-methods), 4
profBin, 26, 52, 83, 84
profBinLin, 83
profBinLinBase, 83
profinfo, 81
profinfo (xcmsSet-class), 84
profinfo, xcmsRaw-method
(xcmsRaw-class), 80
profinfo, xcmsSet-method
(xcmsSet-class), 84
profinfo<- (xcmsSet-class), 84
profinfo<-, xcmsSet-method
(xcmsSet-class), 84
profIntLin, 83
profMedFilt, 81
profMedFilt (profMedFilt-methods), 52
profMedFilt, xcmsRaw-method
(profMedFilt-methods), 52
profMedFilt-methods, 52
profMethod, 52, 54, 79, 81, 84
profMethod (profMethod-methods), 52
profMethod, xcmsRaw-method
(profMethod-methods), 52
profMethod, xcmsSet-method
(xcmsSet-class), 84
profMethod-methods, 52
profMethod<-, 81
profMethod<-, (profMethod-methods), 52
profMethod<-, xcmsRaw-method
(profMethod-methods), 52
profMz (xcmsRaw-class), 80
profMz, xcmsRaw-method (xcmsRaw-class),
80
profRange, 32, 45, 50, 81
profRange (profRange-methods), 53
profRange, xcmsRaw-method
(profRange-methods), 53
profRange-methods, 53
profStep, 79, 82, 84
profStep (profStep-methods), 54
profStep, xcmsRaw-method
(profStep-methods), 54
profStep, xcmsSet-method
(xcmsSet-class), 84
profStep-methods, 54
profStep<-, 82
profStep<-, (profStep-methods), 54
profStep<-, xcmsRaw-method
(profStep-methods), 54
progressCallback (xcmsSet-class), 84
progressCallback, xcmsSet-method
(xcmsSet-class), 84
progressCallback<-, xcmsSet-method
(xcmsSet-class), 84
progressCallback<-, xcmsSet-method
(xcmsSet-class), 84
INDEX

rawEIC, 30, 46
rawEIC (rawEIC-methods), 54
rawEIC, xcmsRaw-method (rawEIC-methods), 54
rawEIC-methods, 54
rawMat (rawMat-methods), 55
rawMat, xcmsRaw-method (rawMat-methods), 55
rawMat-methods, 55
retcor, 49, 85
retcor (retcor-methods), 56
retcor, xcmsSet-method (retcor-methods), 56
retcor-methods, 56
retcor.linear (retcor.peakgroups-methods), 57
retcor.linear, xcmsSet-method (retcor.peakgroups-methods), 57
retcor.loess, 56
retcor.loess (retcor.peakgroups-methods), 57
retcor.loess, xcmsSet-method (retcor.peakgroups-methods), 57
retcor.obiwarp, 56, 56, 58
retcor.obiwarp, xcmsSet-method (retcor.obiwarp), 56
retcor.peakgroups (retcor.peakgroups-methods), 57
retcor.peakgroups, xcmsSet-method (retcor.peakgroups-methods), 57
retcor.peakgroups-methods, 57
retexp, 58
revMz (xcmsRaw-class), 80
revMz, xcmsRaw-method (xcmsRaw-class), 80
rtrange (xcmsEIC-class), 72
rtrange, xcmsEIC-method (xcmsEIC-class), 72
sampclass, 4, 83
sampclass (xcmsSet-class), 84
sampclass, xcmsSet-method (xcmsSet-class), 84
sampclass<- (xcmsSet-class), 84
sampclass<- (xcmsSet-class), 84
sampnames, 72, 86
sampnames (sampnames-methods), 59
sampnames, xcmsEIC-method (sampnames-methods), 59
sampnames, xcmsSet-method (sampnames-methods), 59
sampnames-methods, 59
sampnames<- (xcmsSet-class), 84
sampnames<- (xcmsSet-class), 84
scanrange (xcmsSet-class), 84
scanrange, xcmsRaw-method (xcmsRaw-class), 80
scanrange, xcmsSet-method (xcmsSet-class), 84
selfStart, 65
SerialParam, 83
show, 75
show, xcmsEIC-method (xcmsEIC-class), 72
show, xcmsFragments-method (xcmsFragments-class), 75
show, xcmsPeaks-method (xcmsPeaks-class), 77
show, xcmsRaw-method (xcmsRaw-class), 80
show, xcmsSet-method (xcmsSet-class), 84
SnowParam, 83
sortMz (xcmsRaw-class), 80
sortMz, xcmsRaw-method (xcmsRaw-class), 80
specDist (specDist-methods), 59
specDist, xcmsSet-method (specDist-methods), 59
specDist-methods, 59
specDist.cosine, 60
specDist.cosine, matrix, matrix-method (specDist.cosine), 60
specDist.meanMZmatch, 61
specDist.meanMZmatch, matrix, matrix-method (specDist.meanMZmatch), 61
specDist.peakCount (specDist.peakCount-methods), 62
specDist.peakCount, matrix, matrix-method (specDist.peakCount-methods), 62
specDist.peakCount-methods, 62
specNoise, 62, 63
specPeaks, 63, 63
split, 86
split, split-methods (split.xcmsSet), 64
split.screen, 42, 46
split.xcmsRaw, 64
split.xcmsSet, 64
SSgauss, 65
stitch (stitch-methods), 66
stitch, xcmsRaw-method (stitch-methods), 66
stitch-methods, 66
stitch.netCDF (stitch-methods), 66
stitch.xml (stitch-methods), 66
structure, 78
updateObject, xcmsSet-method, 67
valueCount2ScanIndex, 67
vector, 78
verify.mzQuantM, 68
verify.mzQuantML, 70
verify.mzQuantML (verify.mzQuantM), 68
write.cdf (write.cdf-methods), 69
write.cdf, xcmsRaw-method
  (write.cdf-methods), 69
write.cdf-methods, 69
write.mzdata (write.mzdata-methods), 69
write.mzdata, xcmsRaw-method
  (write.mzdata-methods), 69
write.mzdata-methods, 69
write.mzQuantML, 68
write.mzQuantML
  (write.mzQuantML-methods), 70
write.mzQuantML, xcmsSet-method
  (write.mzQuantML-methods), 70
write.mzQuantML-methods, 70
writeMzTab, 71
xcms-deprecated, 72
xcmsEIC-class, 72
xcmsFileSource, 86
xcmsFileSource-class, 73
xcmsFragments, 7, 74, 75, 79
xcmsFragments-class, 75
xcmsPapply, 76, 84
xcmsPeaks-class, 77
xcmsRaw, 7, 25, 32, 39, 41, 69, 70, 76, 78, 80, 82, 86, 87
xcmsRaw-class, 80
xcmsSet, 7, 25, 47, 70, 71, 75, 82, 84–86
xcmsSet-class, 84
xcmsSource, 40, 73, 87
xcmsSource (xcmsSource-methods), 87
xcmsSource, character-method
  (xcmsFileSource-class), 73
xcmsSource, xcmsSource-method
  (xcmsSource-methods), 87
xcmsSource-class, 86
xcmsSource-methods, 87