xcms
November 11, 2009

R topics documented:

calibrate-methods ................................................. 2
collect-methods .................................................. 3
colMax ............................................................... 4
c-methods ........................................................... 5
descendZero ......................................................... 6
diffreport-methods .................................................. 7
doubleMatrix ........................................................ 9
etg ................................................................. 9
fillPeaks.chrom-methods ......................................... 10
fillPeaks-methods .................................................. 11
fillPeaks.MSW-methods ............................................ 11
filtft ................................................................. 12
findEqualGreater ................................................... 13
findPeaks.centWave-methods ...................................... 14
findPeaks.matchedFilter-methods ................................. 15
findPeaks-methods .................................................. 17
findPeaks.MS1-methods ............................................. 18
findPeaks.MSW-methods ............................................ 19
getEIC-methods ..................................................... 20
getPeaks-methods .................................................. 21
getScan-methods .................................................... 22
getSpec-methods .................................................... 22
group-methods ...................................................... 23
group-methods ...................................................... 24
group.mzClust ....................................................... 24
grounpnames-methods ............................................. 25
group-methods ...................................................... 26
groupval-methods ................................................... 27
image-methods ....................................................... 27
medianFilter ........................................................ 28
na.flatfill ........................................................... 29
netCDF ............................................................... 29
panel.cor ............................................................ 30
plotChrom-methods ................................................ 31
plotPeaks-methods ................................................ 32
plotRaw-methods ................................................... 32
plotrt-methods ...................................................... 33
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

wishlist  
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

wishlist  
for each sample can be used a different wishlist if there is given a list of m/z-vectors. 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-wishlist and a shift outside.

Methods

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

See Also

xcmsSet-class,

Description

Collecting Peaks into xcmsFragments from several MS-runs using xcmsSet and xcmsRaw.

Arguments

object  
(empty) xcmsFragments-class object

xs  
A xcmsSet-class object which contains picked ms1-peaks from several experiments

compMethod  
("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.

snthresh, mzgap, uniq  
these are the parameters for the getspec-peakpicker included in xcmsRaw.
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

A matrix with columns:

- **peakID**: unique identifier of every peak
- **MSnParentPeakID**: PeakID of the parentpeak of a `msLevel>`1 - peak, it is 0 if the peak is `msLevel` 1.
- **msLevel**: The `msLevel` of the peak.
- **rt**: retention time of the peak midpoint
- **mz**: the `mz`-Value of the peak
- **intensity**: the intensity of the peak
- **sample**: the number of the sample from the `xcmsSet`

Methods

```
object = "xcmsFragments"  collect(object, ...)
```

---

**colMax**

Find row and column maximum values

Description

Find row and column maximum values for numeric arrays.

Usage

```
colMax(x, na.rm = FALSE, dims = 1)
rowMax(x, na.rm = FALSE, dims = 1)
which.colMax(x, na.rm = FALSE, dims = 1)
which.rowMax(x, na.rm = FALSE, dims = 1)
```

Arguments

- **x**: an array of two or more dimensions, containing numeric values
- **na.rm**: logical. Should missing values (including 'NaN') be omitted from the calculations? (not currently implemented)
- **dims**: Which dimensions are regarded as "rows" or "columns" to maximize. For `rowMax`, the maximum is over dimensions `dims+1, ...`; for `colMax` it is over dimensions `1:dims`.

Details

These functions are designed to act like the `colSums` series of functions except that they only currently handle real arrays and will not remove NA values.
Value

A numeric array of suitable size, or a vector if the result is one-dimensional. The dimnames (or names for a vector result) are taken from the original array.

For the which.* functions, an integer array of suitable size, or a vector if the result is one-dimensional. The indecies returned are for accessing x one-dimensionally (i.e. x[index]). For which.colMax(), the actual row indecies may be determined using (which.colMax(x)-1) %% nrow(x) + 1. For which.rowMax(), the actual column indecies may be determined using ceiling(rowMax(x)/nrow(x)).

Author(s)

Colin A. Smith, ⟨csmith@scripps.edu⟩

See Also

colSums

---

**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"` c(xs1, ...)

Author(s)

Colin A. Smith, ⟨csmith@scripps.edu⟩

See Also

xcmsSet-class
descendZero  

Find start and end points of a peak

Description

Descends down the sides of a data peak and finds either the points greater than or equal to the zero intercept, the intercept with a given value, or the bottom of the first valley on each side.

Usage

```r
descendZero(y, istart = which.max(y))
descendValue(y, value, istart = which.max(y))
descendMin(y, istart = which.max(y))
```

Arguments

- `y` numeric vector with values
- `istart` starting point for descent
- `value` numeric value to descend to

Value

An integer vector of length 2 with the starting and ending indicies of the peak start and end points.

Author(s)

Colin A. Smith, ⟨csmith@scripps.edu⟩

See Also

descendValue

Examples

```r
normdist <- dnorm(seq(-4, 4, .1)) - .1
cms:::descendZero(normdist)
normdist[cms:::descendZero(normdist)]
cms:::descendValue(normdist, .15)
normdist[cms:::descendValue(normdist, .15)]
cms:::descendMin(normdist)
```
diffreport-methods  Create report of analyte differences

Description

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.
- **...**: 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 on 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 groups, `xcms` will produce boxplots of the different groups 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 class 2, negative for analytes having greater intensity in class 1
- **pvalue**: p-value of t-statistic
- **anova**: p-value of the ANOVA statistic if there are multiple groups
- **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
  ...
  one column for every sample class
- **Sample Names**: integrated intensity value for every sample
  ...
  one column for every sample

### Methods

```r
object = "xcmsSet"  diffreport(object, class1 = levels(sampclass(object))[1],
class2 = levels(sampclass(object))[2], filebase = character(), eicmax = 0, eicwidth = 200, sortpval = TRUE, classeic = c(class1,class2),
value=c("into","maxo","intb"), metlin = FALSE, h=480,w=640, ...)
```

### See Also

`xcmsSet-class, mt.teststat, palette`
**doubleMatrix**

Allocate double, integer, or logical matrices in one step without copying memory around.

**Usage**

```r
doubleMatrix(nrow = 0, ncol = 0)
integerMatrix(nrow = 0, ncol = 0)
logicalMatrix(nrow = 0, ncol = 0)
```

**Arguments**

- `nrow`: number of matrix rows
- `ncol`: number of matrix columns

**Value**

Matrix of double, integer, or logical values. Memory is not zeroed.

**Author(s)**

Colin A. Smith, ⟨csmith@scripps.edu⟩

---

**etg**

*Empirically Transformed Gaussian function*

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
Integrate areas of missing peaks

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

\[
\begin{align*}
\text{object} & \quad \text{the xcmsSet object} \\
\end{align*}
\]

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.

Methods

\[
\text{object} = \"xcmsSet\" \quad \text{fillPeaks.chrom(object)}
\]

See Also
xcmsSet-class, getPeaks fillPeaks
fillPeaks-methods

Integrate areas of missing peaks

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`

fillPeaks.MSW-methods

Integrate areas of missing peaks in FTICR-MS data

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
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`

---

**filtfft**

Apply an convolution filter using an FFT

Description

Expands a vector to the length of the filter and then convolutes it using two successive FFTs.

Usage

```r
filtfft(y, filt)
```

Arguments

- `y`: numeric vector of data to be filtered
- `filt`: filter with length `nextn(length(y))`

Value

A numeric vector the same length as `y`.

Author(s)

Colin A. Smith, (`csmith@scripps.edu`)
findEqualGreater

Find values in sorted vectors

Description

Find values in sorted vectors.

Usage

findEqualGreater(x, value)
findEqualLess(x, value)
findEqualGreaterM(x, values)
findRange(x, values, NAOK = FALSE)

Arguments

x numeric vector sorted in increasing order
value value to find in x
values numeric values to find in x
NAOK don’t check for NA values in x

Details

findEqualGreater finds the index of the first value in x that is equal or greater than value. findEqualLess does same except that it finds equal or less. findEqualGreaterM creates an index of a vector by finding specified values. findRange locates the start and stop indicides of a range of two x values.

The only things that save time at this point are findeEqualGreaterM (when the length of values approaches the length of x) and findRange (when NAOK is set to TRUE). They run in log(N) and N time, respectively.

Value

An integer vector with the position(s) of the values(s).

Author(s)

Colin A. Smith, (csmith@scripps.edu)
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 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.
inintegrate 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

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
findPeaks.matchedFilter-methods

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.
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
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

object = "xcmsRaw" findPeaks.centWave(object, ppm=25, peakwidth=c(20,50), snthresh=10, prefilter=c(3,100), integrate=1, mzdiff=-0.001, fitgauss=FALSE, scanrange= numeric(), noise=0, sleep=0, verbose.columns=FALSE)

Author(s)
Ralf Tautenhahn, ⟨rtautenh@ipb-halle.de⟩

See Also

findPeaks-methods xcmsRaw-class

findPeaks.matchedFilter-methods

Feature detection in the chromatographic time domain

Description
Find peaks in extracted the chromatographic time domain of the profile matrix.
Arguments

- **object**: xcmsRaw object
- **fwhm**: full width at half maximum of matched filtration gaussian model peak
- **sigma**: standard deviation 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

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
- **intf**: integrated area of filtered peak
- **maxo**: maximum intensity of original (raw) peak
- **maxf**: maximum intensity of filtered peak
- **i**: rank of peak identified in merged EIC (<= max)
- **sn**: signal to noise ratio of the peak

Methods

```r
object = "xcmsRaw"  findPeaks.matchedFilter(object, fwhm = 30, sigma = fwhm/2.3548, max = 5, snthresh = 10, step = 0.1, steps = 2, mzdiff = 0.8 - step*steps, index = FALSE, sleep = 0)
```

Author(s)

Colin A. Smith, (csmith@scripps.edu)

See Also

- `findPeaks-methods`
- `xcmsRaw-class`
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
rmin leading edge of peak retention time
rmax 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 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:  
<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:  
<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).

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

object = "xcmsRaw" findPeaks.MS1(object)
Feature detection for single-spectrum non-chromatography MS data

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.

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**

```r
object = "xcmsRaw"  findPeaks.MSW(object, snthresh=3, scales=seq(1,22,3),
nearbyPeak=TRUE, peakScaleRange=5, amp.Th=0.01, minNoiseLevel=amp.Th/SNR.Th,
ridgeLength=24, tuneIn=FALSE, sleep=0, verbose.columns = FALSE)
```

**Author(s)**

Steffen Neumann, Joachim kutzer, ⟨sneumann|jkutzer@ipb-halle.de⟩

**See Also**

`findPeaks-methods` `xcmsRaw-class` `peakDetectionCWT`

---

**getEIC-methods**  
*Get extracted ion chromatograms for specified m/z ranges*

**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.

**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` for `xcmsSet` objects, if left blank the group data will be used instead

- `rtrange`  
  A two column matrix the same size as `mzrange` with minimum and maximum retention times between which to return EIC data points for `xcmsSet` objects, it may also be a single number specifying the time window around the peak to return EIC data points

- `step`  
  Step size to use for profile generation

- `groupidx`  
  Either character vector with names or integer vector with indicies of peak groups for which to get EICs

- `sampleidx`  
  Either character vector with names or integer vector with indicies of samples for which to get EICs

- `rt`  
  "corrected" for using corrected retention times, or "raw" for using raw retention times

**Value**

For `xcmsRaw` objects, if `rtrange` is NULL, an intensity matrix with a row for each `mzmin`, `mzmax` pair. Columns correspond to individual scans. If `rtrange` is not NULL, a list of two column (retention time/intensity) matricies, one for each `mzmin, mzmax` pair.

For `xcmsSet` objects, an `xcmsEIC` object.
getPeaks-methods

Methods

object = "xcmsRaw" getEIC(object, mzrange, rrange = NULL, step = 0.1)
object = "xcmsSet" getEIC(object, mzrange, rrange = 200, groupidx, sampleidx = sampnames(object), rt = c("corrected", "raw"))

See Also

xcmsRaw-class, xcmsSet-class, xcmsEIC-class

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 (<= 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

object = "xcmsRaw" getPeaks(object, peakrange, step = 0.1)

See Also

xcmsRaw-class
getScan-methods

Get m/z and intensity values for a single mass scan

Description

Return the data from a single mass scan using the numeric index of the scan as a reference.

Arguments

- **object**: the `xcmsRaw` object
- **scan**: integer index of scan. If negative, the index numbered from the end
- **massrange**: limit data points returned to those between in the range, `range(massrange)`

Value

A matrix with two columns:

- **mz**: m/z values
- **intensity**: intensity values

Methods

```r
object = "xcmsRaw" getScan(object, scan, massrange = numeric())
```

See Also

`xcmsRaw-class.getSpec`

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.
A matrix with two columns:

- \textbf{mz} \: \textit{m/z values}
- \textbf{intensity} \: \textit{intensity values}

\textbf{Methods}

\begin{itemize}
  \item \texttt{object = "xcmsRaw"} \texttt{getSpec(object, ...)}
\end{itemize}

\textbf{See Also}

\texttt{xcmsRaw-class, profRange, getScan}

---

\textbf{Description}

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

\textbf{Arguments}

\begin{itemize}
  \item \textbf{object} \: the \texttt{xcmsSet} object
  \item \textbf{minfrac} \: minimum fraction of samples necessary in at least one of the sample groups for it to be a valid group
  \item \textbf{minsamp} \: minimum number of samples necessary in at least one of the sample groups for it to be a valid group
  \item \textbf{bw} \: bandwidth (standard deviation or half width at half maximum) of gaussian smoothing kernel to apply to the peak density chromatogram
  \item \textbf{mzwid} \: width of overlapping m/z slices to use for creating peak density chromatograms and grouping peaks across samples
  \item \textbf{max} \: maximum number of groups to identify in a single m/z slice
  \item \textbf{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.
\end{itemize}

\textbf{Value}

An \texttt{xcmsSet} object with peak group assignments and statistics.

\textbf{Methods}

\begin{itemize}
  \item \texttt{object = "xcmsSet"} \texttt{group(object, bw = 30, minfrac = 0.5, minsamp = 1, mzwid = 0.25, max = 50, sleep = 0)}
\end{itemize}

\textbf{See Also}

\texttt{xcmsSet-class, density}
**group-methods**

*Group peaks from different samples together*

**Description**

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

**Arguments**

- `object` *xcmsSet-class object*
- `method` Method to use for grouping. 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 density-based approach described by Smith et al (2006) one would use: `group(object, method="density")`. 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")$xcms$group.methods`. If the nickname of a method is called "mzClust", the help page for that specific method can be accessed with `?group.mzClust`.

**Value**

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

**Methods**

`object = "xcmsSet"` `group(object, ...)`

**See Also**

`group.density group.mzClust xcmsSet-class`

**group.mzClust**

*Group Peaks via High Resolution Alignment*

**Description**

Runs high resolution alignment on the peaks stored in a given `xcmsSet`.

**Usage**

`groupedobject <- group(object, method="mzClust", mzppm = 20, mzabs = 0, minsamp`
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.

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
## load xcms and faahKO Dataset
library(xcms)
library(faahKO)
data(faahko)
## group faahko data
##faahko_grouped <- group(faahko, method="mzClust")
## get groups
##groups <- groups(faahko_grouped)
## get grouped peak indices
##idx <- groupidx(faahko_grouped)
```

Description

Generate unique names for peak groups

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

group-methods

Group peaks from different samples together

Description

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

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

See Also
xcmsSet-class, group.density and group.mzClust
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 with rows for every group and columns for every sample. Missing peaks have `NA` values.

**Methods**

- `object = "xcmsSet"`  
  `groupval(object, method = c("medret", "maxint"), value = "index", intensity = "into")`

**See Also**

- `xcmsSet-class`

---

image-methods  

*Plot log intensity image of a xcmsRaw object*

**Description**

Create log intensity false-color image of a `xcmsRaw` object plotted with m/z and retention time axes

**Arguments**

- **x**  
  `xcmsRaw` object
- **col**  
  vector of colors to use for for the image
- **...**  
  arguments for `profRange`

**Methods**

- `x = "xcmsRaw"`  
  `image(x, col = rainbow(256), ...)"`
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

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

Examples

```r
mat <- matrix(1:25, nrow=5)
mat
mmedianFilter(mat, 1, 1)
```
**na.flatfill**  
*Fill in NA values at the extremes of a vector*

**Description**

Extend the first and last real values in a vector to fill in any NA values present.

**Usage**

```r
na.flatfill(x)
```

**Arguments**

- `x` numeric vector with NA values

**Value**

Modified vector.

**Author(s)**

Colin A. Smith, ⟨csmith@scripps.edu⟩

---

**netCDF**  
*High-performance, low-level access to NetCDF data files*

**Description**

Functions for low-level access to numerical data in NetCDF files. They have been optimized for maximum speed and memory efficiency.

**Usage**

```r
netCDFStrError(ncerr)  
netCDFIsFile(filename)  
netCDFOpen(filename)  
netCDFClose(ncid)  
netCDFVarID(ncid, var)  
netCDFVarLen(ncid, var)  
netCDFVarDouble(ncid, var)  
netCDFVarInt(ncid, var)  
netCDFMSPtrPoints(ncid, scanIndex)  
netCDFRawData(ncid)
```

**Arguments**

- `filename` path to NetCDF file
- `ncid` NetCDF file ID
- `var` NetCDF variable ID or variable name
- `scanIndex` double vector with `scan_index` from CDF file
Details

The mechanisms and structure of NetCDF files are not discussed here, please see the NetCDF documentation for more information.

Value

- `netCDFStrError` returns the textual description of an error code.
- `netCDFIsFile` returns `TRUE` if the referenced file is a NetCDF file and `FALSE` otherwise.
- `netCDFOpen` returns the `ncid` of the opened file.
- `netCDFClose` returns the closing status.
- `netCDFVarID` returns the variable ID.
- `netCDFVarLen` returns the variable length.
- `netCDFVarDouble` and `netCDFVarInt` return either double or integer numeric vectors.
- `netCDFMSPoints` returns a list with elements `massValues` and `intensityValues`, with the masses for each scan in ascending order.
- `netCDFRawData` returns a named list with components `rt`, `tic`, `scanindex`, `mz`, and `intensity`.

An error in any of the functions will return an integer error code with attribute `errortext` containing the error description.

Author(s)

Colin A. Smith, ⟨csmith@scripps.edu⟩

References

NetCDF file format: [http://my.unidata.ucar.edu/content/software/netcdf/](http://my.unidata.ucar.edu/content/software/netcdf/)

---

**panel.cor**

*Correlation coefficient panel for pairs function*

**Description**

Correlation coefficient panel for pairs function.

**Usage**

`panel.cor(x, y, digits = 2, prefix = "", cex.cor)`

**Arguments**

- `x`  
  first data series
- `y`  
  second data series
- `digits`  
  number of digits to plot
- `prefix`  
  text to prefix the coefficients
- `cex.cor`  
  character expansion factor
Author(s)

Colin A. Smith, ⟨csmith@scripps.edu⟩, based on pairs example code

See Also

pairs

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 indecies 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

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

See Also

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

See Also
- `xcmsRaw-class`, `findPeaks`, `split.screen`

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
plotrt-methods

Value

A matrix with the points plotted.

Methods

object = "xcmsRaw" plotRaw(object, mzrange = numeric(), rtrange = numeric(), scanrange = numeric(), log=FALSE, title='Raw Data')

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.

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
- `massrange`: numeric vector of length >= 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, massrange = 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), ...)
plotSurf-methods  
Plot profile matrix 3D surface using OpenGL

Description
This method uses the rgl package to create interactive threedimensional 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

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
ident         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

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
groupidx   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
profGenerate 37

Value

A `xcmsSet` object.

Methods

```R
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`

---

profGenerate  Generation of profile data

Description

Generates profile data in a given range from an indexed pair of vectors.

Usage

```R
profBin(x, y, num, xstart = min(x), xend = max(x), param = list())
profBinM(x, y, zidx, num, xstart = min(x), xend = max(x), NAOK = FALSE, param = list())
profBinLin(x, y, num, xstart = min(x), xend = max(x), param = list())
profBinLinM(x, y, zidx, num, xstart = min(x), xend = max(x), NAOK = FALSE, param = list())
profBinLinBase(x, y, num, xstart = min(x), xend = max(x), param = list())
profBinLinBaseM(x, y, zidx, num, xstart = min(x), xend = max(x), NAOK = FALSE, param = list())
profIntLin(x, y, num, xstart = min(x), xend = max(x), param = list())
profIntLinM(x, y, zidx, num, xstart = min(x), xend = max(x), NAOK = FALSE, param = list())
profMaxIdx(x, y, num, xstart = min(x), xend = max(x), param = list())
profMaxIdxM(x, y, zidx, num, xstart = min(x), xend = max(x), NAOK = FALSE, param = list())
```

Arguments

- `x`: numeric vector of value positions
- `y`: numeric vector of values to bin
- `zidx`: starting position of each new segment
- `num`: number of equally spaced x bins
- `xstart`: starting x value
- `xend`: ending x value
- `NAOK`: allow NA values (faster)
- `param`: parameters for profile generation
Details

These functions take a vector of unequally spaced $y$ values and transform them into either a vector or matrix, depending on whether there is an index or not. Each point in the vector or matrix represents the data for the point centered at its corresponding $x$ value, plus or minus half the $x$ step size $(x_{end} - x_{start})/(num-1)$.

The Bin functions set each matrix or vector value to the maximal point that gets binned into it. The BinLin functions do the same except that they linearly interpolate values into which nothing was binned.

The BinLinBase functions do the same except that they populate empty parts of spectra with a base value. They take to two parameters: 1) baselevel, the intensity level to fill in for empty parts of the spectra. It defaults to half of the minimum intensity. 2) basespace, the $m/z$ length after which the signal will drop to the base level. Linear interpolation will be used between consecutive data points falling within $2 \times$ basespace of each other. It defaults to 0.075.

The IntLin functions set each matrix or vector value to the integral of the linearly interpolated data from plus to minus half the step size.

The MaxIdx functions work similarly to the Bin functions except that the return the integer index of which $x,y$ pair would be placed in a particular cell.

Value

For prof*, a numeric vector of length num.
For prof*M, a matrix with dimensions num by length(zidx).
For MaxIdx, the data type is integer, for all others it is double.

Author(s)

Colin A. Smith, (csmith@scripps.edu)

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**

```r
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
- `massrange` single numeric mass or vector of masses
- `timerange` 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, `massrange`, `scanrange`, and `timerange` can be specified in short form using `m=`, `s=`, and `t=`, respectively. If both a `scanrange` and `timerange` are specified, then the `timerange` 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:

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

Methods

- **object = "xcmsRaw"** profRange(object, massrange = numeric(), timerange = numeric(), scanrange = numeric(), ...)

See Also

- `xcmsRaw-class`

---

**profStep-methods**  
*Get and set m/z step for generating profile data*

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

- **object = "xcmsRaw"** profStep(object)

See Also

- `xcmsRaw-class, profMethod`
**pval**  
*Generate p-values for a vector of t-statistics*

**Description**  
Generate p-values for a vector of Welch’s two-sample t-statistics based on the t distribution.

**Usage**  
```
pval(X, classlabel, teststat)
```

**Arguments**  
- `X`: original data matrix  
- `classlabel`: integer vector with classlabel  
- `teststat`: numeric vector with Welch’s two-sample t-statistics

**Value**  
A numeric vector of p-values.

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

**See Also**  
`mt.teststat`

---

**ramp**  
*High-performance, low-level access to mzXML/mzData files*

**Description**  
Functions for low-level access to data in mzXML/mzData files. They have been optimized for maximum speed and memory efficiency.

**Usage**  
```
rampInit()
rampPrintFiles()
rampIsFile(filename)
rampOpen(filename)
rampClose(rampid)
rampCloseAll()
rampNumScans(rampid)
rampScanHeaders(rampid)
rampSIPeaks(rampid, seqNum, peaksCount)
rampRawData(rampid)
```
Arguments

filename: path to mzXML/mzData file
rampid: RAMP file ID
seqNum: integer vector with indices of scans to read
peaksCount: integer vector with number of peaks in each scan to read

Details

The mechanisms and structure of mzXML/mzData files are not discussed here, please see the mzXML/mzData documentation for more information. These functions make use of the RAMP (Random Access Minimal Parser) code from the SASHIMI open-source project.

rampInit initializes the data structures used for holding RAMP file pointers and indices.
rampPrintFiles prints out a list of all files currently open.
rampCloseAll closes all mzXML/mzData files and frees memory associated with their indices.

Value

rampIsFile returns TRUE if the referenced file is an mzXML or mzData file and FALSE otherwise.
rampOpen returns the rampid of the opened file.
rampClose returns the closing status.
rampNumScans returns the number of scans with an msLevel of 1.
rampSIPeaks returns a named list with components scanindex, mz, and intensity.
rampScanHeaders returns a data frame with header information for each scan.
rampRawData returns a named list with components rt, tic, scanindex, mz, and intensity.

An error in any of the functions will return a (negative) integer error code.

Author(s)

Colin A. Smith, ⟨csmith@scripps.edu⟩

References

mzXML file format: http://sashimi.sourceforge.net/software_glossolalia.html
mzData file format: http://psidev.sourceforge.net/ms/index.html
rawEIC-methods

Get extracted ion chromatograms for specified m/z range

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.

Arguments

| object   | xcmsRaw object |
| massrange | m/z range for EIC |
| scanrange | scan range for EIC |

Value

A list of:

| scan    | scan number |
| intensity | added intensity values |

Methods

object = "xcmsRaw" rawEIC(object, massrange, scanrange=c(1,length(object@scantime))

Author(s)

Ralf Tautenhahn, ⟨rtautenh@ipb-halle.de⟩

See Also

xcmsRaw-class

rawMat-methods

Get a raw data matrix

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

\[
\text{object} = "\text{xcmsRaw}"
\]
\[
\text{rawMat(object, mzrange = numeric(), rtrange = numeric(),}
\]
\[
\text{scanrange = numeric(), log=FALSE)}
\]

Author(s)
Michael Lawrence

See Also

plotRaw for plotting the raw intensities

---

**rectUnique**

**Determine a subset of rectangles with unique, non-overlapping areas**

Description
Given a matrix of rectangular areas, this function determines a subset of those rectangles that do not overlap. Rectangles are preserved on a first come, first served basis, with user control over the order in which the rectangles are processed.

Usage

\[
\text{rectUnique(m, order = seq(length = nrow(m)), xdiff = 0, ydiff = 0)}
\]

Arguments

- \(m\): four column matrix defining rectangular areas
- \(order\): order in which matrix columns should be scanned
- \(xdiff\): maximum space between overlapping rectangles in x dimension
- \(ydiff\): maximum space between overlapping rectangles in y dimension

Details
The \(m\) matrix must contain four columns defining the position of rectangle sides in the following order: left, right, bottom, top. This function is currently implemented in C using a an algorithm with quadratic running time.

Value
A logical vector indicating which rows should be kept.

Author(s)
Colin A. Smith, (csmith@scripps.edu)
Examples

```r
m <- rbind(c(0,4,0,3), c(1,3,2,6), c(3,6,4,6))
plot(0, 0, type = "n", xlim=range(m[,1:2]), ylim=range(m[,3:4]))
rect(m[,1], m[,3], m[,2], m[,4])
xcms:::rectUnique(m)
# Changing order of processing
xcms:::rectUnique(m, c(2,1,3))
# Requiring border spacing
xcms:::rectUnique(m, ydiff = 1)
# Allowing adjacent boxes
xcms:::rectUnique(m, c(2,1,3), xdiff = -0.00001)
# Allowing interpenetration
xcms:::rectUnique(m, xdiff = -1.00001, ydiff = -1.00001)
```

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`. 
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 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
- profStep: step size (in m/z) to use for profile generation from the raw data files
- r: Responsiveness of warping. 0 will give a linear warp based on start and end points. 100 will use all bijective anchors
- g: Gap penalty given in comma separated string: initiate,extend (*Default: (gap_init,gap_extend) [by score type]: 'cor' = '0,3,2.4' 'cov' = '0,11.7' 'prd' = '0,7.8' 'euc' = '0.9,1.8'
- cor: Score function: (*cor (Pearson’s R) or cor_opt (better runtime), cov (covariance), prd (product), euc (Euclidean distance)
- l: Local rather than *global alignment
- i_: Penalty for initiating alignment (for local alignment only) (*Default: 0

Value

An xcmsSet object

Methods

object = "xcmsSet" retcor(object, method="obiwarp", plottype = c("none", "deviation", "mdevden"),col = NULL, ty = NULL, profStep=1, r=NULL, g=NULL, cor = NULL, l=NULL, i_=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 correction groups
- **method**: 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, method = 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
**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**

```r
texp(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`

---

**Get sample names**

**Description**
Return sample names for an object

**Value**
A character vector with sample names.

**Methods**

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

**See Also**
- `xcmsSet-class`, `xcmsEIC-class`
Description

A similarity scoring function for MS/MS spectra against a reference via correlation.

Usage

```r
score_fun.cor(ref, exp)
```

Arguments

- `ref`: An array of numbers for the reference
- `exp`: An array of numbers for the test

Details

A score system using correlation analysis to correlate two arrays. If the arrays are a different length then the system will buffer the shorter one with 0s.

Value

- `score`: Correlation between the two arrays

Author(s)

H. Paul Benton, ⟨hpbenton@scripps.edu⟩

References


See Also

- `score_fun.distMatrix`

Examples

```r
## Not run:
score<-xcms:::score_fun.cor(rnorm(10), rnorm(10))
score

a<-abs(rnorm(5))
a[2]<-xcms:::ppmDev(a[2], 30)
score<-xmcs:::score_FUN.cor()
score
## End(Not run)
```
score_fun.distMatrix

Scoring for MS/MS spectra

Description
A similarity scoring function for MS/MS spectra against a reference via a distance matrix.

Usage

```r
score_fun.distMatrix(ref, exp, ppmfrag)
```

Arguments

- `ref` An array of numbers for the reference
- `exp` An array of numbers for the test
- `ppmfrag` A numerical string for the amount of error in

Details
A simple scoring function to score two arrays of numbers and give a percentage match between the two. Uses a a distance and similarity matrix score system. When the two scores are calculated the percentage score is calculated from the theoritical maximum score and the theoritical minimum score.

Value

- `score` Percentage score between the two arrays

Author(s)
H. Paul Benton, ⟨hpbenton@scripps.edu⟩

References

See Also

- `score_fun.cor`

Examples

```r
## Not run:
score<-xcms:::score_fun.distMatrix(rnorm(10), rnorm(10), 20)
score

a<-abs(rnorm(5))
a[2]<-xcms:::ppmDev(a[2], 30)
score<-xmcs:::score_fun.distMatrix()
score
## End(Not run)
```
**score_fun**  

Scoring for MS/MS spectra

---

**Description**

A similarity scoring function for MS/MS spectra against a reference.

**Usage**

`score_fun(ref, exp, method="distMatrix", ...)`

**Arguments**

- `ref`  
  An array of numbers for the reference

- `exp`  
  An array of numbers for the test

- `method`  
  A string of either 'distMatrix' or 'cor'

- `...`  
  Any other arguments to be passed to other functions

**Details**

This is a starting method to score MS/MS data. Which reports the parameters to the various one of the scoring functions.

**Value**

- `score`  
  Percentage score between the two arrays

**Author(s)**

H. Paul Benton, ⟨hpbenton@scripps.edu⟩

**References**

DOI:http://pubs.acs.org/doi/abs/10.1021/ac800795f/

**See Also**

`score_fun.distMatrix, score_fun.cor`
searchMetlin  

Search Metlin Online Database

Description

A method for searching $M^2S^2$ data against the accurate $M^2S^2$ METLIN database

Usage

```r
searchMetlin(object, ppmfrag=10, ppmMZ= 5, file, MS1data=FALSE, metXML="metlin", ...)
```

Arguments

- **object**: An xcmsFragment object generated by xcmsRaw.collect
- **ppmfrag**: Error in ppm for each fragment
- **ppmMZ**: Error in ppm for precursor mass
- **file**: Name of the results file
- **MS1data**: Should accurate mass alone be search as well
- **metXML**: Location of metlin like XML file or "metlin" as a default
- **...**: Arguments to plot.metlin()

Details

This method automates the task of MS/MS comparison to a reference library. By default the METLIN database is used however this can be changed with the `metXML` parameter.

The search first identifies precursors that match entries in the current METLIN databass using the specified error given by `ppmMZ`. Once a matching m/z value is found, MS/MS data is searched. Each fragment is identified and compared to the reference fragments with error specified by `ppmfrag`. Each match is done using a score schema of the difference and similarity of the two spectra. This value is the equiovacated against the possible maximum and minimum.

For each match a plot of the two MS/MS spectra are given. These are found using the A and B parameter

Value

A data frame with the following columns:

- **A**: Location of the plot
- **B**: Seconded number locator for plot
- **Precursor Ion**: M/Z of the precursor Ion
- **rtmin**: Start of rt window
- **rtmax**: End of rt window
- **CollisionEnergy experiment**: Collision energy of the experiment
- **CollisionEnergy Reference**: Collision energy of the reference
**simSearch**

Percentage Match
- Match percentage of the reference spectra to the experimental spectra

Metlin Mass
- The mass of the reference precursor ion

# matching
- The number of matching fragments

# non-matching
- The number of non-matching fragments

Total # Ref ion
- The total number of fragment reference ions

Metlin ID Name
- Name of the identified metabolite

Ionization
- Is the reference spectra in '-' mode or '+' mode

Adduct
- Is the reference spectra an adduct of the precursor

**Author(s)**

H. Paul Benton, ⟨hpbenton@scripps.edu⟩

**References**


---

**simSearch**  
*Unrestricted precursor Metlin Search*

**Description**

A method for searching $MS^2$ data against the METLIN Database without a precursor restraint.

**Usage**

`simSearch(object, ppmfrag=20, percent=50, file, fullReport=FALSE, ...)`

**Arguments**

- **object**  
  - An xcmsFragment object generated by xcmsRaw.collect

- **ppmfrag**  
  - Error on fragment masses in ppm

- **percent**  
  - Percentage threshold to use for identification

- **file**  
  - Name of the output files

- **fullReport**  
  - Should a full report be generated

  ...  
  - Arguments to plot.metlin()
Details

This method searches the METLIN database for similar MS/MS spectra and ranks them on a fragment score and a neutral loss score. Both of these scores work on a score_fun method. The search takes xcmsFragment objects and searches the database with an unrestricted precursor, thereby searching all spectra in the METLIN database. The percent variable can be used to remove spectra that are below the accepted percentage similarity. The percentage similarity is an independent variable for both the fragment search and the neutral loss search. The method generates two files when the fullReport variable is set to TRUE. The default file shows the top 5 m/z’s most frequently matched. This gives a guide as fragments and neutral losses which should be inspected with a formula calculator. The second file which is printed to a variable and or to the second file is a full report of the data. This report includes metabolite names from the METLIN database and gives both the fragment score and the neutral loss score thereby giving a confidence to the likelihood of the possible molecule/family of molecules.

Value

A data frame with the following columns:

- m/z: Precursor m/z of the Experimental spectra
- rtmin: Start of the rt window
- rtmax: End of the rt window
- Experiment Collision Energy: Experimental spectra Collision Energy
- Fragment Score: Score of the Fragments
- Neutral Score: Score of the Neutral loss
- Common Neutral loss: m/z of the most matching neutral loss
- Common Fragment: m/z of the most matching fragment
- Compound Name: Name of the compound from METLIN
- Metlin Mass: The mass as reported by METLIN
- Collision Energy: The collision energy of the metlin spectra

Author(s)

H. Paul Benton, ⟨hpbenton@scripps.edu⟩

References

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.

Usage

```r
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

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`

specPeaks

Identify peaks in a sparse continuum mode spectrum

Description

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

Usage

```r
specPeaks(spec, sn = 20, mzgap = 0.2)
```
split.xcmsRaw

Arguments

spec matrix with named columns m/z 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 \textasciitilde n 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

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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xs</td>
<td>xcmsSet object</td>
</tr>
<tr>
<td>f</td>
<td>factor such that factor(f) defines the grouping</td>
</tr>
<tr>
<td>drop</td>
<td>logical indicating if levels that do not occur should be dropped (if ’f’ is a ‘factor’ or a list).</td>
</tr>
<tr>
<td>...</td>
<td>further potential arguments passed to methods.</td>
</tr>
</tbody>
</table>

Value

A list of xcmsSet objects.

Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xs = &quot;xcmsSet&quot;</td>
<td>split(x, f, drop = TRUE, ...)</td>
</tr>
</tbody>
</table>

Author(s)

Colin A. Smith, ⟨csmith@scripps.edu⟩

See Also

xcmsSet-class
SSgauss  

**Gaussian Model**

**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**

\[
SSgauss(x, \mu, \sigma, h)
\]

**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`
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`.

---

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.

Author(s)

Colin A. Smith, ⟨csmith@scripps.edu⟩

See Also

- getEIC

---

**xcmsFragments-class**

*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 colms 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 as reported by XXX), sample (The ID of the rawData-file).

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 peaktable 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.
xcmsFragments

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

xcmsFragments Constructor 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

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
Description
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
```r
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/](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

http://ace.acadiau.ca/math/ACMMaC/software/papply/

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

Extends


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-class**

Class xcmsRaw, a class for handling raw data

**Description**

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**

- **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
- **tic**: numeric vector with total ion count (intensity) for each scan
- **scantime**: numeric vector with acquisition time (in seconds) for each scan
- **scanindex**: integer vector with starting positions of each scan in the `mz` and `intensity` variables (note that index values are based off a 0 initial position instead of 1)
- **profmethod**: character value with name of method used for generating the profile matrix
- **mzrange**: numeric vector of length 2 with minimum and maximum m/z values represented in the profile matrix
- **gradient**: matrix with first row, `time`, containing the time point for interpolation and successive columns representing solvent fractions at each point
- **msmsinfo**: matrix with first row, `parent`, containing parent ion m/z and successive columns representing additional information about fragmentation (energy, etc.)

**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
- **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
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"): plot a mass spectrum from profile data
plotTIC signature(object = "xcmsRaw"): experimental method for plotting 3D surface of profile data with rgl.
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
profStep<- signature(object = "xcmsRaw"): change the m/z step used for generating the profile matrix
profStep signature(object = "xcmsRaw"): get the m/z step used for generating the profile matrix
revMz signature(object = "xcmsRaw"): reverse the order of the data points for each scan
sortMz signature(object = "xcmsRaw"): sort the data points by increasing m/z for each scan

Note
No notes yet.

Author(s)
Colin A. Smith, ⟨csmith@scripps.edu⟩

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

See Also
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 (maxrix) mode for efficient plotting and data exploration.

Usage
xcmsRaw(filename, profstep = 1, profmethod = "intlin", profparam = list(), includeMSn=FALSE)

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)

Details
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.)

Value
A xcmsRaw object.

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

References
NetCDF file format: http://my.unidata.ucar.edu/content/software/netcdf/
http://www.astm.org/Standards/E2077.htm
http://www.astm.org/Standards/E2078.htm
mzXML file format: http://sashimi.sourceforge.net/software_glossolalia.html
PSI-MS working group who developed mzData and mzXML file formats: http://www.psidev.info/index.php?q=node/80

See Also
xcmsRaw-class, profStep, profMethod xcmsFragments
xCMSset-class

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.

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
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 two values, method - profile generation method, and step - profile m/z step size

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
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
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
retcor signature(object = "xcmsSet"): use initial grouping of peaks to do nonlinear loess retention time correction
sampclass <- signature(object = "xcmsSet"): DEPRECATED. If used, the experimental design will be replaced with a data frame with a single column matching the supplied factor.
sampclass signature(object = "xcmsSet"): get the interaction of the experimental design factors
phenoData <- signature(object = "xcmsSet"): set the phenoData slot
phenoData signature(object = "xcmsSet"): set the phenoData 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 into a list of objects

Note
No notes yet.

Author(s)
Colin A. Smith, ⟨csmith@scripps.edu⟩

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

See Also
xcmsSet

xcmsSet Constructor for xcmsSet objects which finds peaks in NetCDF/mzXML files

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.

Usage
xcmsSet(files = NULL, snames = NULL, sclass = NULL, phenoData = NULL, profmethod = "bin", profparam = list(), polarity = NULL, nSlaves=0, ...)

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.

Usage
xcmsSet(files = NULL, snames = NULL, sclass = NULL, phenoData = NULL, profmethod = "bin", profparam = list(), polarity = NULL, nSlaves=0, ...)
Arguments

files           path names of the NetCDF/mzXML files to read
snames         sample names
sclass         sample classes
phenoData      sample names and classes
profmeth       method to use for profile generation
profparam      parameters to use for profile generation
polarity       filter raw data for positive/negative scans
nSlaves        number of MPI-slaves to use for parallel peak detection, works only if Rmpi is installed properly, see xcmsPapply.
...             further arguments to the findPeaks method of the xcmsRaw class

Details

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.

Value

A xcmsSet object.

Author(s)

Colin A. Smith, (csmith@scripps.edu)

See Also

xcmsSet-class, findPeaks, profStep, profMethod
Index

*Topic NA
  na.flatfill, 27
*Topic aplot
  panel.cor, 28
*Topic array
  colMax, 3
  rectUnique, 42
*Topic classes
  xcmsEIC-class, 57
  xcmsFragments-class, 58
  xcmsPeaks-class, 61
  xcmsRaw-class, 62
  xcmsSet-class, 65
*Topic file
  calibrate-methods, 1
  diffreport-methods, 5
  fillPeaks-methods, 9
  fillPeaks.chrom-methods, 8
  fillPeaks.MSW-methods, 10
  getEIC-methods, 18
  group-methods, 21, 24
  group.mzClust, 22
  groupnames-methods, 23
  netCDF, 27
  ramp, 39
  retcor.peakgroups-methods, 45
  sampnames-methods, 46
  write.cdf-methods, 57
  xcmsFragments, 59
  xcmsRaw, 64
  xcmsSet, 66
*Topic hplot
  image-methods, 25
  plot.xcmsEIC, 34
  plotChrom-methods, 29
  plotPeaks-methods, 30
  plotRaw-methods, 30
  plotrt-methods, 31
  plotScan-methods, 32
  plotSpec-methods, 32
  plotSurf-methods, 33
  plotTIC-methods, 33
*Topic internal
  colMax, 3
  descendZero, 4
  doubleMatrix, 7
  filtfft, 10
  findEqualGreater, 11
  na.flatfill, 27
  netCDF, 27
  panel.cor, 28
  profGenerate, 35
  pval, 39
  ramp, 39
  rectUnique, 42
*Topic iplot
  plotChrom-methods, 29
  plotSpec-methods, 32
  plotSurf-methods, 33
  plotTIC-methods, 33
*Topic iteration
  descendZero, 4
*Topic manip
  c-methods, 4
  colMax, 3
  getPeaks-methods, 19
  getScan-methods, 20
  getSpec-methods, 20
  groupval-methods, 25
  medianFilter, 26
  profGenerate, 35
  profMedFilt-methods, 36
  profMethod-methods, 37
  profRange-methods, 37
  profStep-methods, 38
  retexp, 46
  specNoise, 53
  specPeaks, 53
  split.xcmsRaw, 54
  split.xcmsSet, 55
*Topic math
  filtfft, 10
*Topic methods
  calibrate-methods, 1
  collect-methods, 2
  diffreport-methods, 5
INDEX

fillPeaks-methods, 9  
fillPeaks.chrom-methods, 8  
fillPeaks.MSW-methods, 10  
findPeaks-methods, 15  
findPeaks.centWave-methods, 12  
findPeaks.matchedFilter-methods, 13  
findPeaks.MS1-methods, 16  
findPeaks.MSW-methods, 17  
getEIC-methods, 18  
getPeaks-methods, 19  
getScan-methods, 20  
getSpec-methods, 20  
group-methods, 21, 22, 24  
group.mzClust, 22  
groupnames-methods, 23  
groupval-methods, 25  
plot.xcmsEIC, 34  
plotChrom-methods, 29  
plotPeaks-methods, 30  
plotRaw-methods, 30  
plotrt-methods, 31  
plotScan-methods, 32  
plotSpec-methods, 32  
plotSurf-methods, 33  
plotTIC-methods, 33  
profMedFilt-methods, 36  
profMethod-methods, 37  
profRange-methods, 37  
profStep-methods, 38  
rawEIC-methods, 41  
rawMat-methods, 41  
retcor-methods, 43, 44  
retcor.peakgroups-methods, 45  
sampnames-methods, 46  
searchMetlin, 50  
simSearch, 51  
write.cdf-methods, 57

*Topic models
  etg, 7

*Topic nonlinear
  SSGauss, 56

*Topic programming
  doubleMatrix, 7

*Topic univar
  pval, 39

*Topic utilities
  findEqualGreater, 11

array, 61

c, 65

c, c-methods (c-methods), 4

c-methods, 4

c.xcmsSet (c-methods), 4

calibrate (calibrate-methods), 1

etg, 7

filepaths (xcmsSet-class), 65
filepaths, xcmsSet-method (xcmsSet-class), 65
filepaths<- (xcmsSet-class), 65
filepaths<-, xcmsSet-method (xcmsSet-class), 65
fillPeaks, 9, 10, 65
fillPeaks(fillPeaks-methods), 9
fillPeaks, xcmsSet-method (fillPeaks-methods), 9
fillPeaks-methods, 9
fillPeaks.chrom (fillPeaks.chrom-methods), 8

fillPeaks.chrom, xcmsSet-method (fillPeaks.chrom-methods), 8

fillPeaks.chrom-methods, 8
fillPeaks.MSW (fillPeaks.MSW-methods), 10
fillPeaks.MSW, xcmsSet-method (fillPeaks.MSW-methods), 10
INDEX

fillPeaks.MSW-methods, 10
filtfft, 10
findEqualGreater, 11
findEqualGreaterM
  (findEqualGreater), 11
findEqualLess(findEqualGreater), 11
findPeaks, 19, 30, 37, 61, 62, 67
findPeaks(findPeaks-methods), 15
findPeaks,xcmsRaw-method
  (findPeaks-methods), 15
findPeaks-methods, 13, 14, 17, 18
findPeaks-methods, 19
findPeaks.centWave, 5, 15
findPeaks.centWave
  (findPeaks.centWave-methods), 12
findPeaks.centWave,xcmsRaw-method
  (findPeaks.centWave-methods), 12
findPeaks.matchedFilter, 15
findPeaks.matchedFilter
  (findPeaks.matchedFilter-methods), 13
findPeaks.matchedFilter,xcmsRaw-method
  (findPeaks.matchedFilter-methods), 13
findPeaks.matchedFilter-methods, 13
findPeaks.MS1
  (findPeaks.MS1-methods), 16
findPeaks.MS1,xcmsRaw-method
  (findPeaks.MS1-methods), 16
findPeaks.MS1-methods, 16
findPeaks.MSW
  (findPeaks.MSW-methods), 17
findPeaks.MSW,xcmsRaw-method
  (findPeaks.MSW-methods), 17
findPeaks.MSW-methods, 17
findRange(findEqualGreater), 11
getEIC, 41, 46, 57, 58, 62, 65
getEIC(getEIC-methods), 18
getEIC,xcmsRaw-method
  (getEIC-methods), 18
getEIC,xcmsSet-method
  (getEIC-methods), 18
getEIC-methods, 18
getPeaks, 9, 10, 62
getPeaks(getPeaks-methods), 19
getPeaks,xcmsRaw-method
  (getPeaks-methods), 19
getPeaks-methods, 19
group, 65
group,xcmsSet-method
  (group-methods), 22
group.density, 22, 24
group.density(group-methods), 21
group.density,xcmsSet-method
  (group-methods), 21
group.mzClust, 22, 24
group.mzClust,xcmsSet-method
  (group.mzClust), 22
group.idx(xcmsSet-class), 65
groupidx,xcmsSet-method
  (group-methods), 24
groupidx(nearest(xcmsSet-class), 24
groupidx(nearest,xcmsSet-method
  (group-methods), 24
groupnames, 58, 65
groupnames(groupnames-methods), 23
groupnames,xcmsEIC-method
  (groupnames-methods), 23
groupnames,xcmsSet-method
  (groupnames-methods), 23
groupnames-methods, 23
groups(xcmsSet-class), 65
groups,xcmsSet-method
  (xcmsSet-class), 65
groups<-(xcmsSet-class), 65
groups<-,xcmsSet-method
  (xcmsSet-class), 65
groupval, 65
groupval(groupval-methods), 25
groupval,xcmsSet-method
  (groupval-methods), 25
groupval-methods, 25
image, 63
INDEX

profinfo<-, xcmsSet-method (xcmsSet-class), 65
profIntLin (profGenerate), 35
profIntLinM (profGenerate), 35
profMaxIdx (profGenerate), 35
profMaxIdxM (profGenerate), 35
profMedFilt, 63
profMedFilt (profMedFilt-methods), 36
profMedFilt, xcmsRaw-method (profMedFilt-methods), 36
profMethod, 37, 38, 63, 64, 67
profMethod (profMethod-methods), 37
profMethod, xcmsRaw-method (profMethod-methods), 37
profMethod-methods, 37
profMethod<-, (profMethod-methods), 37
profMethod<-, xcmsRaw-method (profMethod-methods), 37
profMethod<-, 63
profMz (xcmsRaw-class), 62
profMz, xcmsRaw-method (xcmsRaw-class), 62
profRange, 20, 21, 29, 32, 33, 63
profRange (profRange-methods), 37
profRange, xcmsRaw-method (profRange-methods), 37
profRange-methods, 37
profStep, 63, 64, 67
profStep (profStep-methods), 38
profStep, xcmsRaw-method (profStep-methods), 38
profStep-methods, 38
profStep<-, (profStep-methods), 38
profStep<-, xcmsRaw-method (profStep-methods), 38
profStep<-, 63
pval, 39
ramp, 39
rampClose (ramp), 39
rampCloseAll (ramp), 39
rampInit (ramp), 39
rampIsFile (ramp), 39
rampNumScans (ramp), 39
rampOpen (ramp), 39
rampPrintFiles (ramp), 39
rampRawData (ramp), 39
rampScanHeaders (ramp), 39
rampSIPEaks (ramp), 39
rawEIC (rawEIC-methods), 41
rawEIC, xcmsRaw-method (rawEIC-methods), 41
rawEIC-methods, 41
rawMat (rawMat-methods), 41
rawMat, xcmsRaw-method (rawMat-methods), 41
rawMat-methods, 41
rectUnique, 42
retcor, 31, 66
retcor (retcor-methods), 43
retcor, xcmsSet-method (retcor-methods), 43
retcor-methods, 43, 44
retcor.linear (retcor.peaks-groups-methods), 45
retcor.linear, xcmsSet-method (retcor.peaks-groups-methods), 45
retcor.loess, 43
retcor.loess (retcor.peaks-groups-methods), 45
retcor.loess, xcmsSet-method (retcor.peaks-groups-methods), 45
retcor.obiwarp, 43, 45
retcor.obiwarp (retcor-methods), 44
retcor.obiwarp, xcmsSet-method (retcor-methods), 44
retcor.peaks-groups (retcor.peaks-groups-methods), 45
retcor.peaks-groups, xcmsSet-method (retcor.peaks-groups-methods), 45
retcor.peaks-groups-methods, 45
retpex, 46
revMz (xcmsRaw-class), 62
revMz, xcmsRaw-method (xcmsRaw-class), 62
rowMax (colMax), 3
rtrange (xcmsEIC-class), 57
rtrange, xcmsEIC-method (xcmsEIC-class), 57
sampclass (xcmsSet-class), 65
sampclass, xcmsSet-method (xcmsSet-class), 65
sampclass<-, (xcmsSet-class), 65
INDEX

sampclass<-, xcmsSet-method (xcmsSet-class), 65
sampnames, 58, 66
sampnames (sampnames-methods), 46
sampnames, xcmsEIC-method (sampnames-methods), 46
sampnames, xcmsSet-method (sampnames-methods), 46
sampnames-methods, 46
sampnames<-, xcmsSet-method (xcmsSet-class), 65
score_fun, 49
score_fun.cor, 47, 48, 49
score_fun.distMatrix, 47, 48, 49
searchMetlin, 50
searchMetlin, xcmsFragments-method (searchMetlin), 50
selfStart, 56
show, 58
show, xcmsEIC-method (xcmsEIC-class), 57
show, xcmsFragments-method (xcmsFragments-class), 58
show, xcmsPeaks-method (xcmsPeaks-class), 61
show, xcmsRaw-method (xcmsRaw-class), 62
show, xcmsSet-method (xcmsSet-class), 65
simSearch, 51
simSearch, xcmsFragments-method (simSearch), 51
sortMz (xcmsRaw-class), 62
sortMz, xcmsRaw-method (xcmsRaw-class), 62
specNoise, 53, 54
specPeaks, 53, 53
split, 66
split, split-methods (split.xcmsSet), 55
split.screen, 30
split.xcmsRaw, 54
split.xcmsSet, 55
SSgauss, 56
structure, 61

vector, 61

which.colMax (colMax), 3
which.rowMax (colMax), 3
write.cdf (write.cdf-methods), 57
write.cdf, xcmsRaw-method (write.cdf-methods), 57
write.cdf-methods, 57
xcmsEIC-class, 19, 24, 35, 46
xcmsEIC-class, 57
xcmsFragments, 2, 58, 59, 64
xcmsFragments-class, 2, 59
xcmsFragments-class, 58
xcmsPapply, 60, 67
xcmsPeaks-class, 61
xcmsRaw, 2, 57, 59, 62, 63, 64
xcmsRaw-class, 13–15, 17–21, 26, 29–34, 36–38, 41, 55, 57, 64
xcmsRaw-class, 62
xcmsSet, 2, 58, 65, 66, 66
xcmsSet-class, 2, 4, 7, 9, 10, 19, 21–25, 31, 43–46, 55, 59, 67
xcmsSet-class, 65