

# xcms

February 8, 2012

---

AutoLockMass-methods

*Automatic parameter for Lock mass fixing 'AutoLockMass' ~~*

---

## Description

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

## Arguments

object            An `xcmsRaw-class` object

## Value

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

## Methods

**object = "xcmsRaw"** signature(object = "xcmsRaw")

## Author(s)

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

## Examples

```
## Not run: library(xcms)
library(faahKO) ## These files do not have this problem to correct for but just for an ex
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
xr<-xcmsRaw(cdffiles[1])
xr
##Lets assume that the lockmass starts at 1 and is every 100 scans
lockMass<-xcms::makeacqNum(xr, freq=100, start=1)
## these are equal
lockmass2<-AutoLockMass(xr)
all((lockmass == lockmass) == TRUE)

ob<-stitch(xr, lockMass)

## End(Not run)
```

---

`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

<code>x</code>	a numeric vector of values at which to evaluate the model
<code>mu</code>	mean of the distribution function
<code>sigma</code>	standard deviation of the distribution function
<code>h</code>	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  $h \cdot \sqrt{2 \cdot \pi}$ .

### Value

A numeric vector of the same length as `x`. It is the value of the expression  $h \cdot \exp(-(x - \mu)^2 / (2 \cdot \sigma^2))$ , 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](mailto:csmith@scripps.edu)>

### See Also

[nls](#), [selfStart](#)

---

absent-methods      *Determine which peaks are absent / present in a sample class*

---

**Description**

Determine which peaks are absent / present in a sample class

**Arguments**

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

**Details**

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

**Value**

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

**Methods**

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

**See Also**

[groupdiffreport](#)

---

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](#)

---

calibrate-methods *Calibrate peaks for correcting unprecise m/z values*

---

**Description**

Calibrate peaks of a `xcmsSet` via a set of known masses

**Arguments**

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

**Value**

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

**Methods**

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

**See Also**

[xcmsSet-class](#),

---

collect-methods      *Collect MS<sup>n</sup> peaks into xcmsFragments*

---

### Description

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

### Arguments

<code>object</code>	(empty) <code>xcmsFragments-class</code> object
<code>xs</code>	A <code>xcmsSet-class</code> object which contains picked ms1-peaks from several experiments
<code>compMethod</code>	("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.
<code>snthresh, mzgap, uniq</code>	these are the parameters for the getspec-peakpicker included in <code>xcmsRaw</code> .

### Details

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

### Value

A matrix with columns:

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

### Methods

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

---

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

<code>object</code>	the <code>xcmsSet</code> object
<code>class1</code>	character vector with the first set of sample classes to be compared
<code>class2</code>	character vector with the second set of sample classes to be compared
<code>filebase</code>	base file name to save report, <code>.tsv</code> file and <code>_eic</code> will be appended to this name for the tabular report and EIC directory, respectively. if blank nothing will be saved
<code>eicmax</code>	number of the most significantly different analytes to create EICs for
<code>eicwidth</code>	width (in seconds) of EICs produced
<code>sortpval</code>	logical indicating whether the reports should be sorted by p-value
<code>classeic</code>	character vector with the sample classes to include in the EICs
<code>value</code>	intensity values to be used for the diffreport. If <code>value="into"</code> , integrated peak intensities are used. If <code>value="maxo"</code> , maximum peak intensities are used. If <code>value="intb"</code> , baseline corrected integrated peak intensities are used (only available if peak detection was done by <code>findPeaks.centWave</code> ).
<code>metlin</code>	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
<code>h</code>	Numeric variable for the height of the eic and boxplots that are printed out.
<code>w</code>	Numeric variable for the width of the eic and boxplots print out made.
<code>...</code>	optional arguments to be passed to <code>mt.teststat</code>

## Details

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

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

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

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

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

### Value

A data frame with the following columns:

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

### Methods

```
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](#)

---

etg

*Empirically Transformed Gaussian function*

---

### Description

A general function for asymmetric chromatographic peaks.

### Usage

```
etg(x, H, t1, tt, k1, kt, lambda1, lambda2, 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
lambda2	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

Jianwei Li. Development and Evaluation of Flexible Empirical Peak Functions for Processing Chromatographic Peaks. *Anal. Chem.*, 69 (21), 4452-4462, 1997. <http://dx.doi.org/10.1021/ac970481d>

---

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 <code>xcmsSet</code> 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

```
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 <code>xcmsSet</code> 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

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

## See Also

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

---

`fillPeaks.chrom-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

## Details

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

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

## Value

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

## Methods

```
object = "xcmsSet" fillPeaks.chrom(object)
```

**See Also**

[xcmsSet-class](#), [getPeaks](#) [fillPeaks](#)

---

 findMZ

*Find fragment ions in xcmsFragment objects*


---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

A data frame with the following columns:

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

**Author(s)**

H. Paul Benton, <hpaul.beonton08@imperial.ac.uk>

**References**

H. Paul Benton, D.M. Wong, S.A. Strauger, G. Siuzdak "XCMS<sup>2</sup>" Analytical Chemistry 2008

**See Also**

[findneutral](#),

**Examples**

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

## End(Not run)
```

---

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**

<code>object</code>	<code>xcmsRaw-class</code> object
<code>method</code>	Method to use for peak detection. See details.
<code>...</code>	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")$xcms$findPeak`. 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:

<code>mz</code>	weighted (by intensity) mean of peak m/z across scans
<code>mzmin</code>	m/z of minimum step
<code>mzmax</code>	m/z of maximum step
<code>rt</code>	retention time of peak midpoint
<code>rtmin</code>	leading edge of peak retention time

rtmax            trailing edge of peak retention time  
 into            integrated area of original (raw) peak  
 maxo            maximum intensity of original (raw) peak  
 and additional columns depending on the chosen 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<sup>n</sup> 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<sup>n</sup> scans) quasi simultaneously, e.g. in data dependent tandem MS or DDIT mode.

Since xcmsFragments attaches *all* MS<sup>n</sup> 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)

**Author(s)**

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

**See Also**

[findPeaks-methods xcmsRaw-class](#)

---

findPeaks.MSW-methods

*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**

```
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 kutzera, <sneumann|jkutzer@ipb-halle.de>

**See Also**

[findPeaks-methods](#) [xcmsRaw-class](#) [peakDetectionCWT](#)

---

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.

mzCenterFun	Function to calculate the m/z center of the feature: wMean intensity weighted mean of the feature m/z values, mean mean of the feature m/z values, apex use m/z value at peak apex, wMeanApex3 intensity weighted mean of the m/z value at peak apex and the m/z value left and right of it, meanApex3 mean of the m/z value at peak apex and the m/z value left and right of it.
integrate	Integration method. If =1 peak limits are found through descent on the mexican hat filtered data, if =2 the descent is done on the real data. Method 2 is very accurate but prone to noise, while method 1 is more robust to noise but less exact.
mzdiff	minimum difference in m/z for peaks with overlapping retention times, can be negative to allow overlap
fitgauss	logical, if TRUE a Gaussian is fitted to each peak
scanrange	scan range to process
noise	optional argument which is useful for data that was centroided without any intensity threshold, centroids with intensity < noise are omitted from ROI detection
sleep	number of seconds to pause between plotting peak finding cycles
verbose.columns	logical, if TRUE additional peak meta data columns are returned

### Details

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

### Value

A matrix with columns:

mz	weighted (by intensity) mean of peak m/z across scans
mzmin	m/z peak minimum
mzmax	m/z peak maximum
rt	retention time of peak midpoint
rtmin	leading edge of peak retention time
rtmax	trailing edge of peak retention time
into	integrated peak intensity
intb	baseline corrected integrated peak intensity
maxo	maximum peak intensity
sn	Signal/Noise ratio, defined as $(\text{maxo} - \text{baseline}) / \text{sd}$ , 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), mzCenterFun="wMean", integrate=1,
  mzdiff=-0.001, fitgauss=FALSE, scanrange= numeric(), noise=0, sleep=0,
  verbose.columns=FALSE)
```

### Author(s)

Ralf Tautenhahn

### References

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

### See Also

[findPeaks-methods xcmsRaw-class](#)

---

findPeaks.massifquant-methods

*Feature detection for high resolution LC/MS data*

---

### Description

Kalman filter based feature detection for high resolution LC/MS data in centroid mode (currently experimental).

### Arguments

object	xcmsRaw object
scanrange	scan range to process scanrange = c(1, lastScan) where lastScan is an integer
minIntensity	All real features should exceed this height.
minCentroids	A lower bound for how many scans a feature spans; a feature only incorporates one centroid per scan

<code>consecMissedLim</code>	As a feature is detected, the Kalman Filter may not find a centroid in every scan; After 1 or more misses, this consecutive missed limit informs massifquant when to stop a Kalman Filter to stop looking for a feature.
<code>criticalVal</code>	<code>criticalVal</code> helps determine the error bounds +/- of the Kalman Filter estimate. If the data has very fine mass resolution, a smaller critical val might be better and vice versa. A centroid apart of the feature should fall within these bounds on any given scan. Much like in the construction of a confidence interval, <code>criticalVal</code> loosely translates to be a multiplier of the standard error estimate reported by the Kalman Filter. It is a relaxed application of the confidence interval because it doesn't change as more data is incorporated into the estimation proces, which would change the degrees of freedom and hence the critical value t.
<code>ppm</code>	maximum m/z deviation in consecutive scans by ppm (parts per million).
<code>segs</code>	( <code>segs = 1</code> #if turned on <code>segs = 0</code> #if turned off) With very few data points, sometimes a Kalman Filter "falls off" and stops tracking a feature prematurely. Another Kalman Filter is instantiated and begins following the rest of the signal. Because tracking is done backwards to forwards, this algorithmic defect leaves a real feature divided into two segments ( <code>segs</code> for segmentation). With this option turned on, the program identifies segmented features and combines them into one with two sample t-test. The only danger is that samples with time consecutive features that appear conjoined to form a saddle will also be combined.
<code>scanBack</code>	( <code>segs = 1</code> #if turned on <code>segs = 0</code> #if turned off) The convergence of a Kalman Filter to a feature's precise m/z mapping is very fast, but sometimes it incorporates erroneous centroids as part of a feature (especially early on). The "scanBack" option removes the occasional outlier that lies beyond the converged bounds of the Kalman Filter. The option does not directly affect identification of a feature because it is a postprocessing measure; nonetheless, can potentially improve the quantitation by removing unlikely elements of an established feature.

## Details

This algorithm is most suitable for high resolution LC/{OrbiTrap, TOF}-MS data in centroid mode. Simultaneous kalman filters identify features and calculate their area under the curve. Originally developed on LTQ Orbitrap data with much less than perfect chromatography, the default parameters are set to that specification. Users will find it useful to do some simple exploratory data analysis to find out where to set a minimum intensity, and identify how many scans an average feature may be. May we suggest using TOPPView as a visualization tool. Historically, the `consecutiveMissedLim` parameter should be set to (2) on Orbitrap data and (1) on TOF data, but never should exceed (4). The `criticalVal` parameter is perhaps most difficult to dial in appropriately and visual inspection of peak identification is the best suggested tool for quick optimization. The `ppm`, `sets`, and `scanBack` parameters have shown less influence than the other parameters and exist to give users flexibility and better accuracy.

## Value

A matrix with columns:

<code>mz</code>	weighted mean (by intensity) of feature m/z across scans
<code>mzmin</code>	m/z peak minimum
<code>mzmax</code>	m/z peak maximum
<code>rt</code>	retention time of peak midpoint estimate

rtmin	leading edge of peak retention time
rtmax	trailing edge of peak retention time
into	integrated peak intensity without any normalization
maxo	maximum peak intensity

## Methods

**object = "xcmsRaw"** For Orbitrap Data with poor to acceptable chromatography, suggested default parameters. `findPeaks.massifquant(object, scanrange = c(1, length(object@scanrange)), minIntensity = 6400, minCentroids = 12, consecMissedLim = 2, criticalVal = 1.7321, ppm = 10, segs = 1, scanBack = 1)`

For TOF Data with perfect chromatography, suggested default parameters. `findPeaks.massifquant(object, scanrange = c(1, length(object@scantime)), minIntensity = 1800, minCentroids = 6, consecMissedLim = 1, criticalVal = 0.7111, ppm = 10, segs = 1, scanBack = 1)`

## Author(s)

Chris Conley

## References

yet another peak finder (still needing a title). High Impact Journal. Nov. 2011.

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

**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**

```
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](#)

---

findneutral

*Find neutral losses in xcmsFragment objects*

---

**Description**

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

**Usage**

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

**Arguments**

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

## Details

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

## Value

A data frame with the following columns:

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

## Author(s)

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

## References

H. Paul Benton, D.M. Wong, S.A. Strauger, G. Siuzdak "XCMS<sup>2</sup>" Analytical Chemistry 2008

## See Also

[findMZ](#),

## Examples

```
## Not run:
library(msdata)
mzdatapath <- system.file("iontrap", package = "msdata")
mzdatafiles <- list.files(mzdatapath, pattern = "extracted.mzData", recursive = TRUE, full.
xs <- xcmsSet(mzdatafiles, method = "MS1")
##takes only one file from the file set
xfrag <- xcmsFragments(xs)
found <- findneutral(xfrag, 58.1455, 50)

## End(Not run)
```

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**

<code>object</code>	the <code>xcmsRaw</code> or <code>xcmsSet</code> object
<code>mzrange</code>	either a two column matrix with minimum or maximum m/z or a matrix of any dimensions containing columns <code>mzmin</code> and <code>mzmax</code> for <code>xcmsSet</code> objects, if left blank the group data will be used instead
<code>rtrange</code>	a two column matrix the same size as <code>mzrange</code> with minimum and maximum retention times between which to return EIC data points for <code>xcmsSet</code> objects, it may also be a single number specifying the time window around the peak to return EIC data points
<code>step</code>	step size to use for profile generation
<code>groupidx</code>	either character vector with names or integer vector with indices of peak groups for which to get EICs
<code>sampleidx</code>	either character vector with names or integer vector with indices of samples for which to get EICs
<code>rt</code>	"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) matrices, one for each `mzmin`, `mzmax` pair.

For `xcmsSet` objects, an `xcmsEIC` object.

**Methods**

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

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

**See Also**

[xcmsRaw-class](#), [xcmsSet-class](#), [xcmsEIC-class](#)

---

getPeaks-methods    *Get peak intensities for specified regions*

---

### Description

Integrate extracted ion chromatograms in pre-defined defined regions. Return output similar to [findPeaks](#).

### Arguments

object	the <code>xcmsSet</code> object
peakrange	matrix or data frame with 4 columns: <code>mzmin</code> , <code>mzmax</code> , <code>rtmin</code> , <code>rtmax</code> (they must be in that order or named)
step	step size to use for profile generation

### Value

A matrix with columns:

<code>i</code>	rank of peak identified in merged EIC ( $\leq$ <code>max</code> ), always NA
<code>mz</code>	weighted (by intensity) mean of peak m/z across scans
<code>mzmin</code>	m/z of minimum step
<code>mzmax</code>	m/z of maximum step
<code>ret</code>	retention time of peak midpoint
<code>retmin</code>	leading edge of peak retention time
<code>retmax</code>	trailing edge of peak retention time
<code>into</code>	integrated area of original (raw) peak
<code>intf</code>	integrated area of filtered peak, always NA
<code>maxo</code>	maximum intensity of original (raw) peak
<code>maxf</code>	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 <code>xcmsRaw</code> object
scan	integer index of scan. if negative, the index numbered from the end
mzrange	limit data points returned to those between in the range, <code>range(mzrange)</code>

### Value

A matrix with two columns:

mz	m/z values
intensity	intensity values

### Methods

**object = "xcmsRaw"** `getScan(object, scan, mzrange = 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 <code>xcmsRaw</code> object
...	arguments passed to <a href="#">profRange</a> used to sepecify the spectral segments of interest for averaging

### Details

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

**Value**

A matrix with two columns:

mz	m/z values
intensity	intensity values

**Methods**

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

**See Also**

[xcmsRaw-class](#), [profRange](#), [getScan](#)

---

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	<a href="#">xcmsSet-class</a> 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.me`. 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.density      *Group peaks from different samples together*

---

### Description

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

### Arguments

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

### Value

An xcmsSet object with peak group assignments and statistics.

### Methods

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

### See Also

[xcmsSet-class](#), [density](#)

---

group.mzClust      *Group Peaks via High Resolution Alignment*

---

### Description

Runs high resolution alignment on single spectra samples stored in a given xcmsSet.

**Arguments**

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

**Value**

Returns a xcmsSet with slots groups and groupindex set.

**Methods**

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

**References**

Saira A. Kazmi, Samiran Ghosh, Dong-Guk Shin, Dennis W. Hill and David F. Grant  
*Alignment of high resolution mass spectra: development of a heuristic approach for metabolomics.*  
 Metabolomics, Vol. 2, No. 2, 75-83 (2006)

**See Also**

[xcmsSet-class](#),

**Examples**

```
## Not run:
library(msdata)
mzdatapath <- system.file("fticr", package = "msdata")
mzdatafiles <- list.files(mzdatapath, recursive = TRUE, full.names = TRUE)

xs <- xcmsSet(method="MSW", files=mzdatafiles, scales=c(1,7), SNR.method='data.mean' , width=1000,
              peakThr=80000, amp.Th=0.005)

xsg <- group(xs, method="mzClust")

## End(Not run)
```

---

group.nearest

*Group peaks from different samples together*

---

**Description**

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

Katajamaa M, Miettinen J, Oresic M: MZmine: Toolbox for processing and visualization of mass spectrometry based molecular profile data. *Bioinformatics* (Oxford, England) 2006, 22:634-636.

Currently, there is no equivalent to minfrac or minsamp.

**Arguments**

<code>object</code>	the <code>xcmsSet</code> object
<code>mzVsRTbalance</code>	Multiplicator for <code>mz</code> value before calculating the (euclidean) distance between two peaks.
<code>mzCheck</code>	Maximum tolerated distance for <code>mz</code> .
<code>rtCheck</code>	Maximum tolerated distance for <code>RT</code> .
<code>kNN</code>	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](#)

groupnames-methods *Generate unique names for peak groups*

**Description**

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

**Arguments**

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

**Value**

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

**Methods**

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

**See Also**

[xcmsSet-class](#), [xcmsEIC-class](#)

---

groupval-methods      *Extract a matrix of peak values for each group*

---

### Description

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

### Arguments

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

<code>x</code>	<code>xcmsRaw</code> object
<code>col</code>	vector of colors to use for for the image
<code>...</code>	arguments for <code>profRange</code>

### Methods

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

**Author(s)**

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

**See Also**

[xcmsRaw-class](#)

---

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

**Author(s)**

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

**Examples**

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

---

peakPlots-methods *Plot a grid of a large number of peaks*

---

### Description

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

### Arguments

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

### Details

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

### Methods

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

### See Also

[xcmsRaw-class](#), [findPeaks](#), [split.screen](#)

---

peakTable-methods *Create report of aligned peak intensities*

---

### Description

Create a report showing all aligned peaks.

### Arguments

object	the <code>xcmsSet</code> object
filebase	base file name to save report, <code>.tsv</code> file and <code>_eic</code> will be appended to this name for the tabular report and EIC directory, respectively. if blank nothing will be saved
...	arguments passed down to <code>groupval</code> , which provides the actual intensities.

## Details

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

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

## Value

A data frame with the following columns:

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

## Methods

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

## See Also

[xcmsSet-class](#),

## Examples

```
## Not run:
library(faahKO)
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
xs<-xcmsSet(cdf files)
xs<-group(xs)
peakTable(xs, filebase="peakList")

## End(Not run)
```

---

plot.xcmsEIC                      *Plot extracted ion chromatograms from multiple files*

---

### Description

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

### Arguments

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

### Value

A xcmsSet object.

### Methods

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

### Author(s)

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

**See Also**

[xcmsEIC-class](#), [png](#), [pdf](#), [postscript](#),

---

plotChrom-methods *Plot extracted ion chromatograms from the profile matrix*

---

**Description**

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

**Arguments**

object	the <code>xcmsRaw</code> 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 <a href="#">profRange</a>

**Value**

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

**Methods**

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

**See Also**

[xcmsRaw-class](#)

---

plotEIC-methods *Plot extracted ion chromatograms for specified m/z range*

---

**Description**

Plot extracted ion chromatogram for *m/z* values of interest. The raw data is used in contrast to [plotChrom](#) which uses data from the profile matrix.

**Arguments**

object	<code>xcmsRaw</code> object
mzrange	<i>m/z</i> range for EIC
rtrange	retention time range for EIC
scanrange	scan range for EIC

**Value**

A two-column matrix with the plotted points.

**Methods**

```
object = "xcmsRaw" plotEIC(object, mzrange = numeric(), rtrange = numeric(),
  scanrange = numeric())
```

**Author(s)**

Ralf Tautenhahn

**See Also**

[rawEIC](#), [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 <code>xcmsRaw</code> object
peaks	matrix with peak information as produced by <a href="#">findPeaks</a>
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 <code>xcmsRaw</code> object
mzrange	numeric vector of length $\geq 2$ whose range will be used to select the masses to plot
rtrange	numeric vector of length $\geq 2$ whose range will be used to select the retention times to plot
scanrange	numeric vector of length $\geq 2$ whose range will be used to select scans to plot
log	logical, log transform intensity
title	main title of the plot

### Value

A matrix with the points plotted.

### Methods

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

### See Also

[xcmsRaw-class](#)

---

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

## Methods

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

## See Also

[xcmsRaw-class](#)

---

plotSpec-methods    *Plot mass spectra from the profile matrix*

---

## Description

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

## Arguments

object	the <code>xcmsRaw</code> object
ident	logical, use mouse to identify and label peaks
vline	numeric vector with locations of vertical lines
...	arguments passed to <a href="#">profRange</a>

## Value

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

## Methods

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

## See Also

[xcmsRaw-class](#)

---

plotSurf-methods     *Plot profile matrix 3D surface using OpenGL*

---

### Description

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

### Arguments

<code>object</code>	the <code>xcmsRaw</code> object
<code>log</code>	logical, log transform intensity
<code>aspect</code>	numeric vector with aspect ratio of the m/z, retention time and intensity components of the plot
<code>...</code>	arguments passed to <a href="#">profRange</a>

### 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

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

**Value**

If `ident == TRUE`, an integer vector with the indices 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](#)

---

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**

<code>object</code>	the <code>xcmsSet</code> object
<code>col</code>	vector of colors for plotting each sample
<code>ty</code>	vector of line and point types for plotting each sample
<code>leg</code>	logical plot legend with sample labels
<code>densplit</code>	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](#)

---

profMedFilt-methods

*Median filtering of the profile matrix*

---

### Description

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

### Arguments

object	the <code>xcmsRaw</code> object
massrad	number of m/z grid points on either side to use for median calculation
scanrad	number of scan grid points on either side to use for median calculation

### Methods

**object = "xcmsRaw"** `profMedFilt(object, massrad = 0, scanrad = 0)`

### See Also

[xcmsRaw-class](#), [medianFilter](#)

---

profMethod-methods *Get and set method for generating profile data*

---

### Description

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

### Methods

**object = "xcmsRaw"** `profMethod(object)`

### See Also

[xcmsRaw-class](#), [profMethod](#), [profBin](#), [plotSpec](#), [plotChrom](#), [findPeaks](#)

---

profRange-methods *Specify a subset of profile mode data*

---

### Description

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

### Arguments

object	the <code>xcmsRaw</code> object
mzrange	single numeric mass or vector of masses
rtrange	single numeric time (in seconds) or vector of times
scanrange	single integer scan index or vector of indecies
...	arguments to other functions

### Details

This function handles selection of mass/time subsets of the profile matrix for other functions. It allows the user to specify such subsets in a variety of flexible ways with minimal typing.

Because R does partial argument matching, `mzrange`, `scanrange`, and `rtrange` can be specified in short form using `m=`, `s=`, and `t=`, respectively. If both a `scanrange` and `rtrange` are specified, then the `rtrange` specification takes precedence.

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

### Value

A list with the folloing items:

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

### Methods

```
object = "xcmsRaw" profRange(object, mzrange = numeric(), rtrange = 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](#)

**Examples**

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

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

profStep(xset)<-0.1 ## decrease the bin size to get better resolution
plotSurf(xset, mass=c(200, 500))
##works nicer on high resolution data.

## End(Not run)
```

---

rawEIC-methods     *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
mzrange	m/z range for EIC
rtrange	retention time range for EIC
scanrange	scan range for EIC

**Value**

A list of :

scan	scan number
intensity	added intensity values

**Methods**

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

**Author(s)**

Ralf Tautenhahn

**See Also**

[xcmsRaw-class](#)

---

rawMat-methods	<i>Get a raw data matrix</i>
----------------	------------------------------

---

**Description**

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

**Arguments**

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

**Value**

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

**Methods**

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

**Author(s)**

Michael Lawrence

**See Also**

[plotRaw](#) for plotting the raw intensities

---

retcor-methods      *Correct retention time from different samples*

---

### Description

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

### Arguments

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

### Details

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

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

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

### Value

An `xcmsSet` object with corrected retention times.

### Methods

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

### See Also

`retcor.loess` `retcor.obiwarp` `xcmsSet-class`,

---

retcor.obiwarp      *Align retention times across samples with Obiwarp*

---

### Description

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

For the original publication see

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

**Arguments**

object	the <code>xcmsSet</code> object
plottype	if deviation plot retention time deviation
profStep	step size (in m/z) to use for profile generation from the raw data files
center	the index of the sample all others will be aligned to. If <code>center==NULL</code> , the sample with the most peaks is chosen as default.
col	vector of colors for plotting each sample
ty	vector of line and point types for plotting each sample
response	Responsiveness of warping. 0 will give a linear warp based on start and end points. 100 will use all bijective anchors
distFunc	DistFunc function: <code>cor</code> (Pearson's R) or <code>cor_opt</code> (default, calculate only 10% diagonal band of distance matrix, better runtime), <code>cov</code> (covariance), <code>prd</code> (product), <code>euc</code> (Euclidean distance)
gapInit	Penalty for Gap opening, see below
gapExtend	Penalty for Gap enlargement, see below
factorDiag	Local weighting applied to diagonal moves in alignment.
factorGap	Local weighting applied to gap moves in alignment.
localAlignment	Local rather than global alignment
initPenalty	Penalty for initiating alignment (for local alignment only) Default: 0 Default gap penalties: (gapInit, gapExtend) [by distFunc type]: 'cor' = '0.3,2.4' 'cov' = '0,11.7' 'prd' = '0,7.8' 'euc' = '0.9,1.8'

**Value**

An `xcmsSet` object

**Methods**

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

**See Also**

[xcmsSet-class](#),

---

`retcor.peakgroups-methods`

*Align retention times across samples*

---

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

**Value**

An `xcmsSet` object

**Methods**

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

**See Also**

[xcmsSet-class](#), [loess](#) [retcor](#).[obiwarp](#)

---

retexp

*Set retention time window to a specified width*

---

**Description**

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

**Usage**

```
retexp(peakrange, width = 200)
```

**Arguments**

peakrange	matrix with columns <code>retmin</code> and <code>retmax</code>
width	new width for the window

**Value**

The altered matrix.

**Author(s)**

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

**See Also**

[getEIC](#)

---

smpnames-methods *Get sample names*

---

**Description**

Return sample names for an object

**Value**

A character vector with sample names.

**Methods**

**object = "xcmsEIC"** smpnames(object)

**object = "xcmsSet"** smpnames(object)

**See Also**

[xcmsSet-class](#), [xcmsEIC-class](#)

---

specDist-methods *Distance methods for xcmsSet, xcmsRaw and xsAnnotate*

---

**Description**

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

**Arguments**

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

**Details**

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

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

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

**Value**

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

**Methods**

**object = "xcmsSet"** specDist(object, peakIDs1, peakIDs2, ...)

**object = "xsAnnotate"** specDist(object, PSpec1, PSpec2, ...)

**Author(s)**

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

---

specDist.cosine     *a Distance function based on matching peaks*

---

**Description**

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

**Usage**

```
specDist.cosine(peakTable1, peakTable2, mzabs=0.001, mzppm=10, mzExp=0.6, intExp
```

**Arguments**

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

**Details**

The result is the cosine-distance of the product from weighted factors of mz and intensity from matching peaks in the two peaktables. The factors are calculated as  $wFact = mz^{mzExp} * int^{intExp}$ . if no distance is calculated (for example because no matching peaks were found) the return-value is NA.

**Methods**

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

**Author(s)**

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

---

specDist.meanMZmatch

*a Distance function based on matching peaks*

---

**Description**

This method calculates the distance of two sets of peaks.

**Usage**

```
specDist.meanMZmatch(peakTable1, peakTable2, matchdist=1, matchrate=1, mzabs=0.0
```

**Arguments**

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

**Details**

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

**Methods**

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

**Author(s)**

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

---

```
specDist.peakCount-methods
```

*a Distance function based on matching peaks*

---

### Description

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

### Usage

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

### Arguments

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

### Methods

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

### Author(s)

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

---

```
specNoise
```

*Calculate noise for a sparse continuum mass spectrum*

---

### Description

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

### Usage

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

### Arguments

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

**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**

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

**Arguments**

spec	matrix with named columns <code>mz</code> and <code>intensity</code>
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<sup>n</sup> data is discarded.

### Arguments

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

### Value

A list of `xcmsRaw` objects.

### Methods

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

### Author(s)

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

### See Also

[xcmsRaw-class](#)

---

split.xcmsSet      *Divide an xcmsSet object*

---

### Description

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

### Arguments

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

**Value**

A list of `xcmsSet` objects.

**Methods**

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

**Author(s)**

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

**See Also**

[xcmsSet-class](#)

---

stitch-methods      *Correct gaps in data*

---

**Description**

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

**Arguments**

<code>object</code>	An <a href="#">xcmsRaw-class</a> object
<code>lockMass</code>	A dataframe of locations of the gaps
<code>freq</code>	The intervals of the lock mass scans
<code>start</code>	The starting lock mass scan location, default is 1

**Details**

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

**Value**

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

**Methods**

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

**Author(s)**

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

**Examples**

```
## Not run: library(xcms)
library(faahKO) ## These files do not have this problem to correct for but just for an ex
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
xr<-xcmsRaw(cdffiles[1])
xr
##Lets assume that the lockmass starts at 1 and is every 100 scans
lockMass<-xcms::makeacqNum(xr, freq=100, start=1)
## these are equal
lockmass<-AutoLockMass(xr)
ob<-stitch(xr, lockMass)
ob

#plot the old data before correction
foo<-rawEIC(xr, m=c(200,210), scan=c(80,140))
plot(foo$scan, foo$intensity, type="h")

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

## End(Not run)
```

---

write.cdf-methods *Save an xcmsRaw object to file*

---

**Description**

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

**Arguments**

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

**Details**

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

**Value**

None.

**Methods**

**object = "xcmsRaw"** write.cdf(object, filename)

**See Also**

[xcmsRaw-class](#), [xcmsRaw](#),

---

`write.mzdata-methods`*Save an `xcmsRaw` object to a file*

---

### Description

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

### Arguments

<code>object</code>	the <code>xcmsRaw</code> object
<code>filename</code>	filename (may include full path) for the mzData file. Pipes or URLs are not allowed.

### Details

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

### Value

None.

### Methods

```
object = "xcmsRaw" write.mzdata(object, filename)
```

### See Also

[xcmsRaw-class](#), [xcmsRaw](#),

---

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

---

### Description

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

### Objects from the Class

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

### Slots

**eic**: list containing named entries for every sample. for each entry, a list of two column EIC matrices with retention time and intensity

**mzrange**: two column matrix containing starting and ending m/z for each EIC

**rtrange**: two column matrix containing starting and ending time for each EIC

**rt**: either "raw" or "corrected" to specify retention times contained in the object

**groupnames**: group names from xcmsSet object used to generate EICs

### Methods

**groupnames** signature(object = "xcmsEIC"): get groupnames slot

**mzrange** signature(object = "xcmsEIC"): get mzrange slot

**plot** signature(x = "xcmsEIC"): plot the extracted ion chromatograms

**rtrange** signature(object = "xcmsEIC"): get rtrange slot

**sampnames** signature(object = "xcmsEIC"): get sample names

### Note

No notes yet.

### Author(s)

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

### See Also

[getEIC](#)

---

xcmsFragments-class

*Class xcmsFragments, a class for handling Tandem MS and MS<sup>n</sup> 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<sup>n</sup> peaks, including the parent ion relationships.

### Objects from the Class

Objects can be created with the [xcmsFragments](#) constructor and filled with peaks using the collect method.

## Slots

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

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

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

## Methods

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

**plotTree** signature(object = "xcmsFragments"): prints a (text based) pseudo-tree of the 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.

## 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<sup>n</sup> 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](#)

---

xcmsPapply

*xcmsPapply*

---

**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**

```
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.

<code>do_trace</code>	If set to TRUE, causes the <code>papply\action</code> function to be traced. i.e. Each statement is output before it is executed by the slaves.
<code>also_trace</code>	If supplied an array of function names, as strings, tracing will also occur for the specified functions.

### Details

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

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

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

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

- OpenMPI : Unix, <http://www.open-mpi.org/>, don't forget to export `MPI_ROOT` before installing `Rmpi` e.g. `export MPI_ROOT=/usr/lib/openmpi`
- DeinoMPI : Windows, <http://mpi.deino.net/>, also see <http://www.stats.uwo.ca/faculty/yu/Rmpi/>

### 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@acadiu.ca>, modified by Ralf Tautenhahn <rtautenh@ipb-halle.de>.

### References

<http://ace.acadiu.ca/math/ACMMaC/software/papply/>

### Examples

```
## 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

Class "[matrix](#)", from data part. Class "[array](#)", by class "matrix", distance 2. Class "[structure](#)", by class "matrix", distance 3. Class "[vector](#)", by class "matrix", distance 4, with explicit coerce.

### 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**

**acquisitionNum:** acquisitionNum

**env:** environment with three variables: `mz` - concatenated m/z values for all scans, `intensity` - corresponding signal intensity for each m/z value, and `profile` - matrix representation of the intensity values with columns representing scans and rows representing equally spaced m/z values

**filepath:** Path to the raw data file

**gradient:** matrix with first row, `time`, containing the time point for interpolation and successive columns representing solvent fractions at each point

**msnAcquisitionNum:** for each scan a unique acquisition number as reported via "spectrum id" (`mzData`) or "<scan num=...>" and "<scanOrigin num=...>" (`mzXML`)

**msnCollisionEnergy:** "CollisionEnergy" (`mzData`) or "collisionEnergy" (`mzXML`)

**msnLevel:** for each scan the "msLevel" (both `mzData` and `mzXML`)

**msnPrecursorCharge:** "ChargeState" (`mzData`) and "precursorCharge" (`mzXML`)

**msnPrecursorIntensity:** "Intensity" (`mzData`) or "precursorIntensity" (`mzXML`)

**msnPrecursorMz:** "MassToChargeRatio" (`mzData`) or "precursorMz" (`mzXML`)

**msnPrecursorScan:** "spectrumRef" (both `mzData` and `mzXML`)

**msnRt:** Retention time of the scan

**msnScanindex:** msnScanindex

**mzrange:** numeric vector of length 2 with minimum and maximum m/z values represented in the profile matrix

**polarity:** polarity

**profmethod:** character value with name of method used for generating the profile matrix

**profparam:** profparam

**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)

**scantime:** numeric vector with acquisition time (in seconds) for each scan

**tic:** numeric vector with total ion count (intensity) for each scan

**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"): experimental method for plotting 3D surface of profile data with rgl.
- plotTIC** signature(object = "xcmsRaw"): plot total ion count chromatogram
- 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
- stitch** signature(object = "xcmsRaw"): Raw data correction for lock mass calibration gaps.

## 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](#)

---

`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 = "bin", profparam =  
list(), includeMSn=FALSE, mslevel=NULL)
```

```
deepCopy(object)
```

### Arguments

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

### 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.)

`deepCopy(xraw)` will create a copy of the `xcmsRaw` object with its own copy of m/z and intensity data in `xraw@env`

### 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](http://sashimi.sourceforge.net/software_glossolalia.html)

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

Parser used for XML file formats: <http://tools.proteomecenter.org/wiki/index.php?title=Software:RAMP>

## See Also

[xcmsRaw-class](#), [profStep](#), [profMethod](#) [xcmsFragments](#)

## Examples

```
## Not run:
library(xcms)
library(faahKO)
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
xr<-xcmsRaw(cdffiles[1])
xr
##This gives some information about the file
names(attributes(xr))
## Lets have a look at the structure of the object

str(xr)
##same but with a preview of each slot in the object
##SO... lets have a look at how this works
head(xr@scanindex)
#[1] 0 429 860 1291 1718 2140
xr@env$mz[425:430]
#[1] 596.3 597.0 597.3 598.1 599.3 200.1
##We can see that the 429 index is the last mz of scan 1 therefore...

mz.scan1<-xr@env$mz[(1+xr@scanindex[1]):xr@scanindex[2]]
intensity.scan1<-xr@env$intensity[(1+xr@scanindex[1]):xr@scanindex[2]]
plot(mz.scan1, intensity.scan1, type="h", main=paste("Scan 1 of file", basename(cdffiles[1])))
##the easier way :p
scan1<-getScan(xr, 1)
head(scan1)
plotScan(xr, 1)

## End(Not run)
```

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

## 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"): get the phenoData slot

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

**progressCallback** signature(object = "xcmsSet"): get the progressCallback 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	<i>Constructor for xcmsSet objects which finds peaks in NetCDF/mzXML files</i>
---------	--

---

## 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, lockMassFreq=FALSE,  
        mslevel=NULL, nSlaves=0, progressCallback=NULL,...)
```

## Arguments

files	path names of the NetCDF/mzXML files to read
snames	sample names
sclass	sample classes
phenoData	sample names and classes
profmethod	method to use for profile generation
profparam	parameters to use for profile generation
polarity	filter raw data for positive/negative scans
lockMassFreq	Performs correction for Waters LockMass function
mslevel	perform peak picking on data of given mslevel
nSlaves	number of slaves/cores to be used for parallel peak detection. MPI is used if installed, otherwise the snow package is employed for multicore support.
progressCallback	function to be called, when progressInfo changes (useful for GUIs)
...	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.

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

## Value

A xcmsSet object.

## Author(s)

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

**See Also**

[xcmsSet-class](#), [findPeaks](#), [profStep](#), [profMethod](#), [xcmsPapply](#)

# Index

## \*Topic **classes**

- xcmsEIC-class, [55](#)
- xcmsFragments-class, [56](#)
- xcmsPeaks-class, [60](#)
- xcmsRaw-class, [60](#)
- xcmsSet-class, [64](#)

## \*Topic **file**

- calibrate-methods, [4](#)
- diffreport-methods, [6](#)
- fillPeaks-methods, [9](#)
- fillPeaks.chrom-methods, [10](#)
- fillPeaks.MSW-methods, [9](#)
- getEIC-methods, [22](#)
- group.density, [26](#)
- group.mzClust, [26](#)
- group.nearest, [27](#)
- groupnames-methods, [28](#)
- peakTable-methods, [31](#)
- retcor.peakgroups-methods, [45](#)
- sampnames-methods, [47](#)
- write.cdf-methods, [54](#)
- write.mzdata-methods, [55](#)
- xcmsFragments, [57](#)
- xcmsRaw, [63](#)
- xcmsSet, [67](#)

## \*Topic **hplot**

- image-methods, [29](#)
- plot.xcmsEIC, [33](#)
- plotChrom-methods, [34](#)
- plotPeaks-methods, [35](#)
- plotRaw-methods, [36](#)
- plotrt-methods, [39](#)
- plotScan-methods, [36](#)
- plotSpec-methods, [37](#)
- plotSurf-methods, [38](#)
- plotTIC-methods, [38](#)

## \*Topic **iplot**

- plotChrom-methods, [34](#)
- plotSpec-methods, [37](#)
- plotSurf-methods, [38](#)
- plotTIC-methods, [38](#)

## \*Topic **lockmass**

- AutoLockMass-methods, [1](#)

## \*Topic **manip**

- AutoLockMass-methods, [1](#)
- c-methods, [3](#)
- getPeaks-methods, [23](#)
- getScan-methods, [24](#)
- getSpec-methods, [24](#)
- groupval-methods, [29](#)
- medianFilter, [30](#)
- profMedFilt-methods, [40](#)
- profMethod-methods, [40](#)
- profRange-methods, [41](#)
- profStep-methods, [42](#)
- retexp, [46](#)
- specNoise, [50](#)
- specPeaks, [51](#)
- split.xcmsRaw, [52](#)
- split.xcmsSet, [52](#)
- stitch-methods, [53](#)

## \*Topic **methods**

- absent-methods, [3](#)
- AutoLockMass-methods, [1](#)
- calibrate-methods, [4](#)
- collect-methods, [5](#)
- diffreport-methods, [6](#)
- fillPeaks-methods, [9](#)
- fillPeaks.chrom-methods, [10](#)
- fillPeaks.MSW-methods, [9](#)
- findMZ, [11](#)
- findneutral, [20](#)
- findPeaks-methods, [12](#)
- findPeaks.centWave-methods, [15](#)
- findPeaks.massifquant-methods, [17](#)
- findPeaks.matchedFilter-methods, [19](#)
- findPeaks.MS1-methods, [13](#)
- findPeaks.MSW-methods, [14](#)
- getEIC-methods, [22](#)
- getPeaks-methods, [23](#)
- getScan-methods, [24](#)
- getSpec-methods, [24](#)
- group-methods, [25](#)

- group.density, 26
- group.mzClust, 26
- group.nearest, 27
- groupnames-methods, 28
- groupval-methods, 29
- peakPlots-methods, 31
- peakTable-methods, 31
- plot.xcmsEIC, 33
- plotChrom-methods, 34
- plotEIC-methods, 34
- plotPeaks-methods, 35
- plotRaw-methods, 36
- plotrt-methods, 39
- plotScan-methods, 36
- plotSpec-methods, 37
- plotSurf-methods, 38
- plotTIC-methods, 38
- profMedFilt-methods, 40
- profMethod-methods, 40
- profRange-methods, 41
- profStep-methods, 42
- rawEIC-methods, 42
- rawMat-methods, 43
- retcor-methods, 44
- retcor.obiwarp, 44
- retcor.peakgroups-methods, 45
- samnames-methods, 47
- specDist-methods, 47
- specDist.cosine, 48
- specDist.meanMZmatch, 49
- specDist.peakCount-methods, 50
- stitch-methods, 53
- write.cdf-methods, 54
- write.mzdata-methods, 55
- \*Topic models**
  - etg, 8
- \*Topic nonlinear**
  - SSgauss, 2
- absent (*absent-methods*), 3
- absent, xcmsSet-method (*absent-methods*), 3
- absent-methods, 3
- array, 60
- AutoLockMass (*AutoLockMass-methods*), 1
- AutoLockMass, xcmsRaw-method (*AutoLockMass-methods*), 1
- AutoLockMass-methods, 1
- c, 65
- c, c-methods (*c-methods*), 3
- c-methods, 3
- c.xcmsSet (*c-methods*), 3
- calibrate (*calibrate-methods*), 4
- calibrate, xcmsSet-method (*calibrate-methods*), 4
- calibrate-methods, 4
- collect, 57, 58
- collect (*collect-methods*), 5
- collect, xcmsFragments-method (*collect-methods*), 5
- collect, xcmsRaw-method (*collect-methods*), 5
- collect-methods, 5
- deepCopy (*xcmsRaw*), 63
- deepCopy, xcmsRaw-method (*xcmsRaw*), 63
- density, 26
- diffreport, 3, 32, 65
- diffreport (*diffreport-methods*), 6
- diffreport, xcmsSet-method (*diffreport-methods*), 6
- diffreport-methods, 6
- etg, 8
- filepaths (*xcmsSet-class*), 64
- filepaths, xcmsSet-method (*xcmsSet-class*), 64
- filepaths<- (*xcmsSet-class*), 64
- filepaths<-, xcmsSet-method (*xcmsSet-class*), 64
- fillPeaks, 3, 10, 11, 65
- fillPeaks (*fillPeaks-methods*), 9
- fillPeaks, xcmsSet-method (*fillPeaks-methods*), 9
- fillPeaks-methods, 9
- fillPeaks.chrom (*fillPeaks.chrom-methods*), 10
- fillPeaks.chrom, xcmsSet-method (*fillPeaks.chrom-methods*), 10
- fillPeaks.chrom-methods, 10
- fillPeaks.MSW (*fillPeaks.MSW-methods*), 9
- fillPeaks.MSW, xcmsSet-method (*fillPeaks.MSW-methods*), 9
- fillPeaks.MSW-methods, 9
- findMZ, 11, 21
- findMZ, xcmsFragments-method (*findMZ*), 11
- findneutral, 12, 20

- findNeutral, xcmsFragments-method  
(*findNeutral*), 20
- findPeaks, 23, 31, 35, 40, 60, 61, 68
- findPeaks (*findPeaks-methods*), 12
- findPeaks, xcmsRaw-method  
(*findPeaks-methods*), 12
- findPeaks-methods, 14, 15, 17, 19, 20
- findPeaks-methods, 12
- findPeaks.centWave, 6, 13
- findPeaks.centWave  
(*findPeaks.centWave-methods*),  
15
- findPeaks.centWave, xcmsRaw-method  
(*findPeaks.centWave-methods*),  
15
- findPeaks.centWave-methods, 15
- findPeaks.massifquant  
(*findPeaks.massifquant-methods*),  
17
- findPeaks.massifquant, xcmsRaw-method  
(*findPeaks.massifquant-methods*),  
17
- findPeaks.massifquant-methods, 17
- findPeaks.matchedFilter, 13
- findPeaks.matchedFilter  
(*findPeaks.matchedFilter-methods*),  
19
- findPeaks.matchedFilter, xcmsRaw-method  
(*findPeaks.matchedFilter-methods*),  
19
- findPeaks.matchedFilter-methods,  
19
- findPeaks.MS1  
(*findPeaks.MS1-methods*), 13
- findPeaks.MS1, xcmsRaw-method  
(*findPeaks.MS1-methods*), 13
- findPeaks.MS1-methods, 13
- findPeaks.MSW  
(*findPeaks.MSW-methods*), 14
- findPeaks.MSW, xcmsRaw-method  
(*findPeaks.MSW-methods*), 14
- findPeaks.MSW-methods, 14
  
- getEIC, 42, 47, 55, 56, 61, 65
- getEIC (*getEIC-methods*), 22
- getEIC, xcmsRaw-method  
(*getEIC-methods*), 22
- getEIC, xcmsSet-method  
(*getEIC-methods*), 22
- getEIC-methods, 22
- getPeaks, 9–11, 61
- getPeaks (*getPeaks-methods*), 23
- getPeaks, xcmsRaw-method  
(*getPeaks-methods*), 23
- getPeaks-methods, 23
- getScan, 25, 61
- getScan (*getScan-methods*), 24
- getScan, xcmsRaw-method  
(*getScan-methods*), 24
- getScan-methods, 24
- getSpec, 24, 51, 61
- getSpec (*getSpec-methods*), 24
- getSpec, xcmsRaw-method  
(*getSpec-methods*), 24
- getSpec-methods, 24
- group, 3, 66
- group (*group-methods*), 25
- group, xcmsSet-method  
(*group-methods*), 25
- group-methods, 25
- group.density, 25, 26, 28
- group.density, xcmsSet-method  
(*group.density*), 26
- group.mzClust, 25, 26, 28
- group.mzClust, xcmsSet-method  
(*group.mzClust*), 26
- group.nearest, 27
- group.nearest, xcmsSet-method  
(*group.nearest*), 27
- groupidx (*xcmsSet-class*), 64
- groupidx, xcmsSet-method  
(*xcmsSet-class*), 64
- groupidx<- (*xcmsSet-class*), 64
- groupidx<- , xcmsSet-method  
(*xcmsSet-class*), 64
- groupnames, 56, 65
- groupnames (*groupnames-methods*),  
28
- groupnames, xcmsEIC-method  
(*groupnames-methods*), 28
- groupnames, xcmsSet-method  
(*groupnames-methods*), 28
- groupnames-methods, 28
- groups (*xcmsSet-class*), 64
- groups, xcmsSet-method  
(*xcmsSet-class*), 64
- groups<- (*xcmsSet-class*), 64
- groups<- , xcmsSet-method  
(*xcmsSet-class*), 64
- groupval, 31, 65
- groupval (*groupval-methods*), 29
- groupval, xcmsSet-method  
(*groupval-methods*), 29
- groupval-methods, 29

- image, [61](#)
- image, `xcmsRaw`-method  
(*image-methods*), [29](#)
- image-methods, [29](#)
- loess, [46](#)
- makeacqNum (*stitch-methods*), [53](#)
- makeacqNum, `xcmsRaw`-method  
(*stitch-methods*), [53](#)
- matrix, [60](#)
- medianFilter, [30](#), [40](#)
- mt.teststat, [6](#), [7](#)
- mzrange (*xcmsEIC-class*), [55](#)
- mzrange, `xcmsEIC`-method  
(*xcmsEIC-class*), [55](#)
- nls, [2](#)
- palette, [7](#)
- pdf, [34](#)
- peakDetectionCWT, [15](#)
- peakPlots, `xcmsSet`-method  
(*peakPlots-methods*), [31](#)
- peakPlots-methods, [31](#)
- peaks (*xcmsSet-class*), [64](#)
- peaks, `xcmsSet`-method  
(*xcmsSet-class*), [64](#)
- peaks<- (*xcmsSet-class*), [64](#)
- peaks<- , `xcmsSet`-method  
(*xcmsSet-class*), [64](#)
- peakTable (*peakTable-methods*), [31](#)
- peakTable, `xcmsSet`-method  
(*peakTable-methods*), [31](#)
- peakTable-methods, [31](#)
- phenoData (*xcmsSet-class*), [64](#)
- phenoData, `xcmsSet`-method  
(*xcmsSet-class*), [64](#)
- phenoData<- (*xcmsSet-class*), [64](#)
- phenoData<- , `xcmsSet`-method  
(*xcmsSet-class*), [64](#)
- plot, [56](#)
- plot, `plot`-methods  
(*plot.xcmsEIC*), [33](#)
- plot.xcmsEIC, [33](#)
- plotChrom, [34](#), [40](#), [61](#)
- plotChrom (*plotChrom-methods*), [34](#)
- plotChrom, `xcmsRaw`-method  
(*plotChrom-methods*), [34](#)
- plotChrom-methods, [34](#)
- plotEIC (*plotEIC-methods*), [34](#)
- plotEIC, `xcmsRaw`-method  
(*plotEIC-methods*), [34](#)
- plotEIC-methods, [34](#)
- plotPeaks (*plotPeaks-methods*), [35](#)
- plotPeaks, `xcmsRaw`-method  
(*plotPeaks-methods*), [35](#)
- plotPeaks-methods, [35](#)
- plotRaw, [43](#), [61](#)
- plotRaw (*plotRaw-methods*), [36](#)
- plotRaw, `xcmsRaw`-method  
(*plotRaw-methods*), [36](#)
- plotRaw-methods, [36](#)
- plotrt, [66](#)
- plotrt (*plotrt-methods*), [39](#)
- plotrt, `xcmsSet`-method  
(*plotrt-methods*), [39](#)
- plotrt-methods, [39](#)
- plotScan, [62](#)
- plotScan (*plotScan-methods*), [36](#)
- plotScan, `xcmsRaw`-method  
(*plotScan-methods*), [36](#)
- plotScan-methods, [36](#)
- plotSpec, [40](#), [62](#)
- plotSpec (*plotSpec-methods*), [37](#)
- plotSpec, `xcmsRaw`-method  
(*plotSpec-methods*), [37](#)
- plotSpec-methods, [37](#)
- plotSurf, [62](#)
- plotSurf (*plotSurf-methods*), [38](#)
- plotSurf, `xcmsRaw`-method  
(*plotSurf-methods*), [38](#)
- plotSurf-methods, [38](#)
- plotTIC, [62](#)
- plotTIC (*plotTIC-methods*), [38](#)
- plotTIC, `xcmsRaw`-method  
(*plotTIC-methods*), [38](#)
- plotTIC-methods, [38](#)
- plotTree (*xcmsFragments-class*), [56](#)
- plotTree, `xcmsFragments`-method  
(*xcmsFragments-class*), [56](#)
- png, [34](#)
- postscript, [34](#)
- present (*absent-methods*), [3](#)
- present, `xcmsSet`-method  
(*absent-methods*), [3](#)
- profBin, [40](#)
- profinfo (*xcmsSet-class*), [64](#)
- profinfo, `xcmsSet`-method  
(*xcmsSet-class*), [64](#)
- profinfo<- (*xcmsSet-class*), [64](#)
- profinfo<- , `xcmsSet`-method  
(*xcmsSet-class*), [64](#)
- profMedFilt, [62](#)
- profMedFilt

- (profMedFilt-methods)*, 40
- profMedFilt, xcmsRaw-method
  - (profMedFilt-methods)*, 40
- profMedFilt-methods, 40
- profMethod, 40, 42, 62, 64, 68
- profMethod (*profMethod-methods*), 40
- profMethod, xcmsRaw-method
  - (profMethod-methods)*, 40
- profMethod-methods, 40
- profMethod<-, 62
- profMethod<-
  - (profMethod-methods)*, 40
- profMethod<-, xcmsRaw-method
  - (profMethod-methods)*, 40
- profMz (*xcmsRaw-class*), 60
- profMz, xcmsRaw-method
  - (xcmsRaw-class)*, 60
- profRange, 24, 25, 34, 37, 38, 62
- profRange (*profRange-methods*), 41
- profRange, xcmsRaw-method
  - (profRange-methods)*, 41
- profRange-methods, 41
- profStep, 62, 64, 68
- profStep (*profStep-methods*), 42
- profStep, xcmsRaw-method
  - (profStep-methods)*, 42
- profStep-methods, 42
- profStep<-, 62
- profStep<- (*profStep-methods*), 42
- profStep<-, xcmsRaw-method
  - (profStep-methods)*, 42
- progressCallback (*xcmsSet-class*), 64
- progressCallback, xcmsSet-method
  - (xcmsSet-class)*, 64
- progressCallback<-
  - (xcmsSet-class)*, 64
- progressCallback<-, xcmsSet-method
  - (xcmsSet-class)*, 64
- rawEIC, 35
- rawEIC (*rawEIC-methods*), 42
- rawEIC, xcmsRaw-method
  - (rawEIC-methods)*, 42
- rawEIC-methods, 42
- rawMat (*rawMat-methods*), 43
- rawMat, xcmsRaw-method
  - (rawMat-methods)*, 43
- rawMat-methods, 43
- retcor, 39, 66
- retcor (*retcor-methods*), 44
- retcor, xcmsSet-method
  - (retcor-methods)*, 44
- retcor-methods, 44
- retcor.linear
  - (retcor.peakgroups-methods)*, 45
- retcor.linear, xcmsSet-method
  - (retcor.peakgroups-methods)*, 45
- retcor.loess, 44
- retcor.loess
  - (retcor.peakgroups-methods)*, 45
- retcor.loess, xcmsSet-method
  - (retcor.peakgroups-methods)*, 45
- retcor.obiwarp, 44, 44, 46
- retcor.obiwarp, xcmsSet-method
  - (retcor.obiwarp)*, 44
- retcor.peakgroups
  - (retcor.peakgroups-methods)*, 45
- retcor.peakgroups, xcmsSet-method
  - (retcor.peakgroups-methods)*, 45
- retcor.peakgroups-methods, 45
- retexp, 46
- revMz (*xcmsRaw-class*), 60
- revMz, xcmsRaw-method
  - (xcmsRaw-class)*, 60
- rtrange (*xcmsEIC-class*), 55
- rtrange, xcmsEIC-method
  - (xcmsEIC-class)*, 55
- sampclass, 3
- sampclass (*xcmsSet-class*), 64
- sampclass, xcmsSet-method
  - (xcmsSet-class)*, 64
- sampclass<- (*xcmsSet-class*), 64
- sampclass<-, xcmsSet-method
  - (xcmsSet-class)*, 64
- sampnames, 56, 66
- sampnames (*sampnames-methods*), 47
- sampnames, xcmsEIC-method
  - (sampnames-methods)*, 47
- sampnames, xcmsSet-method
  - (sampnames-methods)*, 47
- sampnames-methods, 47
- sampnames<- (*xcmsSet-class*), 64
- sampnames<-, xcmsSet-method
  - (xcmsSet-class)*, 64
- selfStart, 2
- show, 57

- show, *xcmsEIC*-method  
(*xcmsEIC*-class), 55
- show, *xcmsFragments*-method  
(*xcmsFragments*-class), 56
- show, *xcmsPeaks*-method  
(*xcmsPeaks*-class), 60
- show, *xcmsRaw*-method  
(*xcmsRaw*-class), 60
- show, *xcmsSet*-method  
(*xcmsSet*-class), 64
- sortMz (*xcmsRaw*-class), 60
- sortMz, *xcmsRaw*-method  
(*xcmsRaw*-class), 60
- specDist (*specDist*-methods), 47
- specDist, *xcmsSet*-method  
(*specDist*-methods), 47
- specDist-methods, 47
- specDist.cosine, 48
- specDist.cosine, matrix, matrix-method  
(*specDist.cosine*), 48
- specDist.meanMZmatch, 49
- specDist.meanMZmatch, matrix, matrix-method  
(*specDist.meanMZmatch*), 49
- specDist.peakCount  
(*specDist.peakCount*-methods),  
50
- specDist.peakCount, matrix, matrix-method  
(*specDist.peakCount*-methods),  
50
- specDist.peakCount-methods, 50
- specNoise, 50, 51
- specPeaks, 51, 51
- split, 66
- split, split-methods  
(*split.xcmsSet*), 52
- split.screen, 31, 35
- split.xcmsRaw, 52
- split.xcmsSet, 52
- SSgauss, 2
- stitch (*stitch*-methods), 53
- stitch, *xcmsRaw*-method  
(*stitch*-methods), 53
- stitch-methods, 53
- stitch.netCDF (*stitch*-methods), 53
- stitch.xml (*stitch*-methods), 53
- structure, 60
- vector, 60
- write.cdf (*write.cdf*-methods), 54
- write.cdf, *xcmsRaw*-method  
(*write.cdf*-methods), 54
- write.cdf-methods, 54
- write.mzdata  
(*write.mzdata*-methods), 55
- write.mzdata, *xcmsRaw*-method  
(*write.mzdata*-methods), 55
- write.mzdata-methods, 55
- xcmsEIC*-class, 22, 28, 34, 47
- xcmsEIC*-class, 55
- xcmsFragments*, 5, 56, 57, 64
- xcmsFragments*-class, 5, 58
- xcmsFragments*-class, 56
- xcmsPapply*, 58, 68
- xcmsPeaks*-class, 60
- xcmsRaw*, 5, 54, 55, 57, 60, 62, 63
- xcmsRaw*-class, 1, 12–15, 17, 19, 20,  
22–25, 30, 31, 34–40, 42, 43, 52–55,  
64
- xcmsRaw*-class, 60
- xcmsSet*, 5, 56, 65, 66, 67
- xcmsSet*-class, 3–5, 7, 9–11, 22, 25–29,  
32, 39, 44–47, 53, 58, 68
- xcmsSet*-class, 64