Package ‘xcms’

October 3, 2017

Version  1.52.0
Date     2017-04-16
Title    LC/MS and GC/MS Data Analysis
Author   Colin A. Smith <csmith@scripps.edu>,
          Ralf Tautenhahn <rtautenh@gmail.com>,
          Steffen Neumann <sneumann@ipb-halle.de>,
          Paul Benton <hpbenton@scripps.edu>,
          Christopher Conley <cjconley@ucdavis.edu>,
          Johannes Rainer <Johannes.Rainer@eurac.edu>
Maintainer Steffen Neumann <sneumann@ipb-halle.de>
Depends  R (>= 2.14.0), methods, Biobase, BiocParallel (>= 1.8.0),
          MSnbase (>= 2.1.10)
Imports  mzR (>= 1.1.6), BiocGenerics, ProtGenerics, lattice,
          RColorBrewer, plyr, RANN, multtest, MassSpecWavelet (>= 1.5.2),
          S4Vectors
Suggests BiocStyle, knitr (>= 1.1.0), faahKO, msdata, ncdf4, rgl,
         microbenchmark, RUnit
Enhances Rgraphviz, Rmpi, XML
Description Framework for processing and visualization of chromatographically
separated and single-spectra mass spectral data. Imports from AIA/ANDI NetCDF,
 mzXML, mzData and mzML files. Preprocesses data for high-throughput, untargeted
analyte profiling.
License  GPL (>= 2) + file LICENSE
URL     http://metlin.scripps.edu/download/ and
         https://github.com/sneumann/xcms
VignetteBuilder knitr
BugReports https://github.com/sneumann/xcms/issues/new
biocViews MassSpectrometry, Metabolomics
RoxygenNote 6.0.1
         ‘do_adjustRtime-functions.R’ ‘functions-binning.R’
         ‘do_findChromPeaks-functions.R’ ‘functions-Params.R’
         ‘do_groupChromPeaks-functions.R’ ‘fastMatch.R’
R topics documented:

'functions-Chromatogram.R' 'functions-utils.R' 'functions-IO.R'
'functions-MSnExp.R' 'functions-ProcessHistory.R'
'functions-xcmsExp.R' 'functions-xcmsFragments.R'
'functions-xcmsRaw.R' 'functions-xcmsSet.R' 'init.R'
'matchpeaks.R' 'methods-Chromatogram.R' 'methods-IO.R'
'methods-MsFeatureData.R' 'methods-OnDiskMSnExp.R'
'methods-Params.R' 'methods-ProcessHistory.R'
'methods-XCMSnExp.R' 'methods-netCdfSource.R'
'methods-rampSource.R' 'methods-xcmsEIC.R'
'methods-xcmsFileSource.R' 'methods-xcmsFragments.R'
'methods-xcmsPeaks.R' 'methods-xcmsRaw.R' 'methods-xcmsSet.R'
'models.R' 'msn2xcmsRaw.R' 'mzClust.R' 'netCDF.R' 'plotQC.R'
'ramp.R' 'specDist.R' 'write.mzquantML.R' 'writemzdata.R'
'writemztab.R' 'xcmsSource.R' 'zzz.R'

NeedsCompilation yes
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>findChromPeaks-centWaveWithPredIsoROIs</td>
<td>63</td>
</tr>
<tr>
<td>findChromPeaks-massifquant</td>
<td>67</td>
</tr>
<tr>
<td>findChromPeaks-matchedFilter</td>
<td>72</td>
</tr>
<tr>
<td>findMZ</td>
<td>77</td>
</tr>
<tr>
<td>findneutral</td>
<td>78</td>
</tr>
<tr>
<td>findPeaks-methods</td>
<td>80</td>
</tr>
<tr>
<td>findPeaks-MSW</td>
<td>81</td>
</tr>
<tr>
<td>findPeaks.addPredictedIsotopeFeatures-methods</td>
<td>85</td>
</tr>
<tr>
<td>findPeaks.centWave-methods</td>
<td>87</td>
</tr>
<tr>
<td>findPeaks.centWaveWithPredictedIsotopeROIs-methods</td>
<td>89</td>
</tr>
<tr>
<td>findPeaks.massifquant-methods</td>
<td>91</td>
</tr>
<tr>
<td>findPeaks.matchedFilter,xcmsRaw-method</td>
<td>94</td>
</tr>
<tr>
<td>findPeaks.MS1-methods</td>
<td>95</td>
</tr>
<tr>
<td>findPeaks.MSW,xcmsRaw-method</td>
<td>97</td>
</tr>
<tr>
<td>GenericParam-class</td>
<td>98</td>
</tr>
<tr>
<td>getEIC-methods</td>
<td>99</td>
</tr>
<tr>
<td>getPeaks-methods</td>
<td>100</td>
</tr>
<tr>
<td>getScan-methods</td>
<td>101</td>
</tr>
<tr>
<td>getSpec-methods</td>
<td>101</td>
</tr>
<tr>
<td>getXcmsRaw-methods</td>
<td>102</td>
</tr>
<tr>
<td>group-methods</td>
<td>103</td>
</tr>
<tr>
<td>group.density</td>
<td>103</td>
</tr>
<tr>
<td>group.mzClust</td>
<td>104</td>
</tr>
<tr>
<td>group.nearest</td>
<td>105</td>
</tr>
<tr>
<td>groupChromPeaks</td>
<td>107</td>
</tr>
<tr>
<td>groupChromPeaks-density</td>
<td>107</td>
</tr>
<tr>
<td>groupChromPeaks-mzClust</td>
<td>111</td>
</tr>
<tr>
<td>groupChromPeaks-nearest</td>
<td>113</td>
</tr>
<tr>
<td>groupnames-methods</td>
<td>116</td>
</tr>
<tr>
<td>groupval-methods</td>
<td>117</td>
</tr>
<tr>
<td>image-methods</td>
<td>118</td>
</tr>
<tr>
<td>imputeLinInterpol</td>
<td>118</td>
</tr>
<tr>
<td>levelplot-methods</td>
<td>120</td>
</tr>
<tr>
<td>loadRaw-methods</td>
<td>121</td>
</tr>
<tr>
<td>medianFilter</td>
<td>122</td>
</tr>
<tr>
<td>MsFeatureData-class</td>
<td>122</td>
</tr>
<tr>
<td>msn2xcmsRaw</td>
<td>130</td>
</tr>
<tr>
<td>peakPlots-methods</td>
<td>131</td>
</tr>
<tr>
<td>peakTable-methods</td>
<td>132</td>
</tr>
<tr>
<td>plot.xcmsEIC</td>
<td>133</td>
</tr>
<tr>
<td>plotAdjustedRtime</td>
<td>134</td>
</tr>
<tr>
<td>plotChrom-methods</td>
<td>135</td>
</tr>
<tr>
<td>plotEIC-methods</td>
<td>136</td>
</tr>
<tr>
<td>plotPeaks-methods</td>
<td>137</td>
</tr>
<tr>
<td>plotQC</td>
<td>137</td>
</tr>
<tr>
<td>plotRaw-methods</td>
<td>138</td>
</tr>
<tr>
<td>plottr-methods</td>
<td>139</td>
</tr>
<tr>
<td>plotScan-methods</td>
<td>140</td>
</tr>
<tr>
<td>plotSpec-methods</td>
<td>140</td>
</tr>
<tr>
<td>plotSurf-methods</td>
<td>141</td>
</tr>
<tr>
<td>plotTIC-methods</td>
<td>141</td>
</tr>
<tr>
<td>ProcessHistory-class</td>
<td>142</td>
</tr>
</tbody>
</table>
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>profMat-xcmsSet</td>
<td>143</td>
</tr>
<tr>
<td>profMedFilt-methods</td>
<td>145</td>
</tr>
<tr>
<td>profMethod-methods</td>
<td>145</td>
</tr>
<tr>
<td>profRange-methods</td>
<td>146</td>
</tr>
<tr>
<td>profStep-methods</td>
<td>147</td>
</tr>
<tr>
<td>rawEIC-methods</td>
<td>147</td>
</tr>
<tr>
<td>rawMat-methods</td>
<td>148</td>
</tr>
<tr>
<td>retcor-methods</td>
<td>149</td>
</tr>
<tr>
<td>retcor.obiwarp</td>
<td>149</td>
</tr>
<tr>
<td>retcor.peaksgroups-methods</td>
<td>150</td>
</tr>
<tr>
<td>retexp</td>
<td>151</td>
</tr>
<tr>
<td>sampnames-methods</td>
<td>152</td>
</tr>
<tr>
<td>showError,xcmsSet-method</td>
<td>152</td>
</tr>
<tr>
<td>specDist-methods</td>
<td>153</td>
</tr>
<tr>
<td>specDist.cosine</td>
<td>154</td>
</tr>
<tr>
<td>specDist.meanMZmatch</td>
<td>155</td>
</tr>
<tr>
<td>specDist.peakscount-methods</td>
<td>155</td>
</tr>
<tr>
<td>specNoise</td>
<td>156</td>
</tr>
<tr>
<td>spec Peaks</td>
<td>157</td>
</tr>
<tr>
<td>split.xcmsRaw</td>
<td>158</td>
</tr>
<tr>
<td>split.xcmsSet</td>
<td>158</td>
</tr>
<tr>
<td>SSgauss</td>
<td>159</td>
</tr>
<tr>
<td>stitch-methods</td>
<td>160</td>
</tr>
<tr>
<td>updateObject,xcmsSet-method</td>
<td>161</td>
</tr>
<tr>
<td>useOriginalCode</td>
<td>161</td>
</tr>
<tr>
<td>verify.mzQuantM</td>
<td>162</td>
</tr>
<tr>
<td>write.cdf-methods</td>
<td>163</td>
</tr>
<tr>
<td>write.mzdata-methods</td>
<td>163</td>
</tr>
<tr>
<td>write.mzQuantML-methods</td>
<td>164</td>
</tr>
<tr>
<td>writeMzTab</td>
<td>165</td>
</tr>
<tr>
<td>xcms-deprecated</td>
<td>166</td>
</tr>
<tr>
<td>xcmsEIC-class</td>
<td>166</td>
</tr>
<tr>
<td>xcmsFileSource-class</td>
<td>167</td>
</tr>
<tr>
<td>xcmsFragments</td>
<td>168</td>
</tr>
<tr>
<td>xcmsFragments-class</td>
<td>169</td>
</tr>
<tr>
<td>xcmsPapply</td>
<td>170</td>
</tr>
<tr>
<td>xcmsPeaks-class</td>
<td>171</td>
</tr>
<tr>
<td>xcmsRaw</td>
<td>172</td>
</tr>
<tr>
<td>xcmsRaw-class</td>
<td>174</td>
</tr>
<tr>
<td>xcmsSet</td>
<td>176</td>
</tr>
<tr>
<td>xcmsSet-class</td>
<td>178</td>
</tr>
<tr>
<td>xcmsSource-class</td>
<td>181</td>
</tr>
<tr>
<td>xcmsSource-methods</td>
<td>181</td>
</tr>
<tr>
<td>[.XCMSnExp,logicalOrNumeric,missing,missing-method</td>
<td>182</td>
</tr>
<tr>
<td>[.xcmsRaw,logicalOrNumeric,missing,missing-method</td>
<td>184</td>
</tr>
</tbody>
</table>

Index 186
absent-methods

Determine which peaks are absent / present in a sample class

Arguments

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

Details

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

Value

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

Methods

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

See Also

group diffreport

adjustRtime

Alignment: Retention time correction methods.

Description

The adjustRtime method(s) perform retention time correction (alignment) between chromatograms of different samples. These methods are part of the modernized xcms user interface.

The implemented retention time adjustment methods are:

- **peakGroups** retention time correction based on alignment of features (peak groups) present in most/all samples. See adjustRtime-peakGroups for more details.
- **obiwarp** alignment based on the complete mz-rt data. This method does not require any identified peaks or defined features. See adjustRtime-obiwarp for more details.

Author(s)

Johannes Rainer
See Also

retcor for the old retention time correction methods. plotAdjustedRtime for visualization of alignment results.

Other retention time correction methods: adjustRtime-obiwarp, adjustRtime-peakGroups

---

### adjustRtime-obiwarp

**Align retention times across samples using Obiwarp**

**Description**

This method performs retention time adjustment using the Obiwarp method [Prince 2006]. It is based on the code at [http://obi-warp.sourceforge.net](http://obi-warp.sourceforge.net) but supports alignment of multiple samples by aligning each against a center sample. The alignment is performed directly on the profile-matrix and can hence be performed independently of the peak detection or peak grouping.

The ObiwarpParam class allows to specify all settings for the retention time adjustment based on the obiwarp method. Class Instances should be created using the ObiwarpParam constructor.

- **binSize**, binSize<-: getter and setter for the binSize slot of the object.
- **centerSample**, centerSample<-: getter and setter for the centerSample slot of the object.
- **response**, response<-: getter and setter for the response slot of the object.
- **distFun**, distFun<-: getter and setter for the distFun slot of the object.
- **gapInit**, gapInit<-: getter and setter for the gapInit slot of the object.
- **gapExtend**, gapExtend<-: getter and setter for the gapExtend slot of the object.
- **factorDiag**, factorDiag<-: getter and setter for the factorDiag slot of the object.
- **factorGap**, factorGap<-: getter and setter for the factorGap slot of the object.
- **localAlignment**, localAlignment<-: getter and setter for the localAlignment slot of the object.
- **initPenalty**, initPenalty<-: getter and setter for the initPenalty slot of the object.

**Usage**

ObiwarpParam(binSize = 1, centerSample = integer(), response = 1L, distFun = "cor_opt", gapInit = numeric(), gapExtend = numeric(), factorDiag = 2, factorGap = 1, localAlignment = FALSE, initPenalty = 0)

```r
## S4 method for signature 'OnDiskMSnExp,ObiwarpParam'
adjustRtime(object, param)
```

```r
## S4 method for signature 'ObiwarpParam'
show(object)
```

```r
## S4 method for signature 'ObiwarpParam'
binSize(object)
```
## S4 replacement method for signature 'ObiwarpParam'
binSize(object) <- value

## S4 method for signature 'ObiwarpParam'
centerSample(object)

## S4 replacement method for signature 'ObiwarpParam'
centerSample(object) <- value

## S4 method for signature 'ObiwarpParam'
response(object)

## S4 replacement method for signature 'ObiwarpParam'
response(object) <- value

## S4 method for signature 'ObiwarpParam'
distFun(object)

distFun(object)

## S4 replacement method for signature 'ObiwarpParam'
distFun(object) <- value

## S4 method for signature 'ObiwarpParam'
gapInit(object)

gapInit(object)

## S4 replacement method for signature 'ObiwarpParam'
gapInit(object) <- value

## S4 method for signature 'ObiwarpParam'
gapExtend(object)

gapExtend(object)

## S4 replacement method for signature 'ObiwarpParam'
gapExtend(object) <- value

## S4 method for signature 'ObiwarpParam'
factorDiag(object)

## S4 replacement method for signature 'ObiwarpParam'
factorDiag(object) <- value

## S4 method for signature 'ObiwarpParam'
factorGap(object)

## S4 replacement method for signature 'ObiwarpParam'
factorGap(object) <- value

## S4 method for signature 'ObiwarpParam'
llocalAlignment(object)

## S4 replacement method for signature 'ObiwarpParam'
llocalAlignment(object) <- value

## S4 method for signature 'ObiwarpParam'
initPenalty(object)

## S4 replacement method for signature 'ObiwarpParam'
initPenalty(object) <- value

## S4 method for signature 'XCMSnExp,ObiwarpParam'
adjustRtime(object, param)

### Arguments

- **binSize**
  - numeric(1) defining the bin size (in mz dimension) to be used for the profile matrix generation. See step parameter in profile-matrix documentation for more details.

- **centerSample**
  - integer(1) defining the index of the center sample in the experiment. It defaults to floor(median(1:length(fileNames(object)))).

- **response**
  - numeric(1) defining the responsiveness of warping with response = 0 giving linear warping on start and end points and response = 100 warping using all bijective anchors.

- **distFun**
  - character defining the distance function to be used. Allowed values are "cor" (Pearson’s correlation), "cor_opt" (calculate only 10% diagonal band of distance matrix; better runtime), "cov" (covariance), "prd" (product) and "euc" (Euclidian distance). The default value is distFun = "cor_opt".

- **gapInit**
  - numeric(1) defining the penalty for gap opening. The default value for gapInit depends on the value of distFun: for distFun = "cor" and distFun = "cor_opt" it is 0.3, for distFun = "cov" and distFun = "prd" 0.0 and for distFun = "euc" 0.9.

- **gapExtend**
  - numeric(1) defining the penalty for gap enlargement. The default value for gapExtend depends on the value of distFun, for distFun = "cor" and distFun = "cor_opt" it is 2.4, for distFun = "cov" 11.7, for distFun = "prd" 1.8 and for distFun = "euc" 7.8.

- **factorDiag**
  - numeric(1) defining the local weight applied to diagonal moves in the alignment.

- **factorGap**
  - numeric(1) defining the local weight for gap moves in the alignment.

- **localAlignment**
  - logical(1) whether a local alignment should be performed instead of the default global alignment.

- **initPenalty**
  - numeric(1) defining the penalty for initiating an alignment (for local alignment only).

- **object**
  - For adjustRtime: an XCMSnExp object.
  - For all other methods: a ObiwarpParam object.

- **param**
  - A ObiwarpParam object containing all settings for the alignment method.

- **value**
  - The value for the slot.

### Value

The ObiwarpParam function returns a ObiwarpParam class instance with all of the settings specified for obiwarp retention time adjustment and alignment.

For adjustRtime, XCMSnExp, ObiwarpParam: a XCMSnExp object with the results of the retention time adjustment step. These can be accessed with the adjustedRtime method. Retention time correction does also adjust the retention time of the identified chromatographic peaks (accessed via...
chromPeaks. Note that retention time correction drops all previous peak grouping results from the result object.

For adjustRtime, OnDiskMSnExp, ObiwarParam: a numeric with the adjusted retention times per spectra (in the same order than rtime).

Slots

- __classVersion__,binSize,centerSample,response,distFun,gapInit,gapExtend,factorDiag,factorGap,localAlignment,initPenalty

See corresponding parameter above. __classVersion__ stores the version from the class.

Slots values should exclusively be accessed via the corresponding getter and setter methods listed above.

Note

These methods and classes are part of the updated and modernized xcms user interface which will eventually replace the retcor methods. All of the settings to the alignment algorithm can be passed with a ObiwarParam object.

Calling adjustRtime on an XCMSnExp object will cause all peak grouping (correspondence) results and any previous retention time adjustment results to be dropped.

Author(s)

Colin Smith, Johannes Rainer

References


See Also

retcor.obiwarp for the old user interface. plotAdjustedRtime for visualization of alignment results.

XCMSnExp for the object containing the results of the alignment.

Other retention time correction methods: adjustRtime-peakGroups, adjustRtime

Examples

library(faahKO)
library(MSnbase)
fls <- dir(system.file("cdf/KO", package = "faahKO"), recursive = TRUE,
full.names = TRUE)

## Reading 2 of the KO samples
raw_data <- readMSData2(fls[1:2])

## Perform retention time correction on the OnDiskMSnExp:
res <- adjustRtime(raw_data, param = ObiwarParam())

## As a result we get a numeric vector with the adjusted retention times for
## all spectra.
We can split this by file to get the adjusted retention times for each file:

```r
resL <- split(res, fromFile(raw_data))
```

Perform retention time correction on an `XCMSnExp`:

```r
mfp <- MatchedFilterParam(synthresh = 20, binSize = 1)
res <- findChromPeaks(raw_data, param = mfp)
res_2 <- adjustRtime(res, param = ObiwarpParam())
```

The retention times of the detected peaks were adjusted.

```
head(rtime(res_2))
head(rtime(raw_data))
```

This method performs retention time adjustment based on the alignment of chromatographic peak groups present in all/most samples (hence corresponding to housekeeping compounds). First the retention time deviation of these peak groups is described by fitting either a polynomial (`smooth = "loess"`) or a linear (`smooth = "linear"`) model to the data points. These models are subsequently used to adjust the retention time of each spectrum in each sample.

The `PeakGroupsParam` class allows to specify all settings for the retention time adjustment based on housekeeping peak groups present in most samples. Instances should be created with the `PeakGroupsParam` constructor.

`adjustRtimePeakGroups` returns the features (peak groups) which would, depending on the provided `PeakGroupsParam`, be selected for alignment/retention time correction.

- `minFraction`, `minFraction<->`: getter and setter for the `minFraction` slot of the object.
- `extraPeaks`, `extraPeaks<->`: getter and setter for the `extraPeaks` slot of the object.
- `smooth`, `smooth<->`: getter and setter for the `smooth` slot of the object.
- `span`, `span<->`: getter and setter for the `span` slot of the object.
- `family`, `family<->`: getter and setter for the `family` slot of the object.
- `peakGroupsMatrix`, `peakGroupsMatrix<->`: getter and setter for the `peakGroupsMatrix` slot of the object.

`adjustRtime`, `XCMSnExp`, `PeakGroupsParam`: performs retention time correction based on the alignment of peak groups (features) found in all/most samples.
Usage

PeakGroupsParam(minFraction = 0.9, extraPeaks = 1, smooth = "loess", 
span = 0.2, family = "gaussian", peakGroupsMatrix = matrix(nrow = 0, 
ncol = 0))

adjustRtimePeakGroups(object, param = PeakGroupsParam())

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

## S4 method for signature 'PeakGroupsParam'
minFraction(object)

## S4 replacement method for signature 'PeakGroupsParam'
minFraction(object) <- value

## S4 method for signature 'PeakGroupsParam'
extraPeaks(object)

## S4 replacement method for signature 'PeakGroupsParam'
extraPeaks(object) <- value

## S4 method for signature 'PeakGroupsParam'
smooth(x)

## S4 replacement method for signature 'PeakGroupsParam'
smooth(object) <- value

## S4 method for signature 'PeakGroupsParam'
span(object)

## S4 replacement method for signature 'PeakGroupsParam'
span(object) <- value

## S4 method for signature 'PeakGroupsParam'
family(object)

## S4 replacement method for signature 'PeakGroupsParam'
family(object) <- value

## S4 method for signature 'PeakGroupsParam'
peakGroupsMatrix(object)

## S4 replacement method for signature 'PeakGroupsParam'
peakGroupsMatrix(object) <- value

## S4 method for signature 'XCMSnExp,PeakGroupsParam'
adjustRtime(object, param)

Arguments

minFraction numeric(1) between 0 and 1 defining the minimum required fraction of samples
in which peaks for the peak group were identified. Peak groups passing this
criteria will aligned across samples and retention times of individual spectra
will be adjusted based on this alignment. For minFraction = 1 the peak group
has to contain peaks in all samples of the experiment.

extraPeaks numeric(1) defining the maximal number of additional peaks for all samples to
be assigned to a peak group (i.e. feature) for retention time correction. For a data
set with 6 samples, extraPeaks = 1 uses all peak groups with a total peak count
<= 6 + 1. The total peak count is the total number of peaks being assigned to
a peak group and considers also multiple peaks within a sample being assigned
to the group.

smooth character defining the function to be used, to interpolate corrected retention
times for all peak groups. Either "loess" or "linear".

span numeric(1) defining the degree of smoothing (if smooth = "loess"). This
parameter is passed to the internal call to loess.

family character defining the method to be used for loess smoothing. Allowed values
are "gaussian" and "symmetric". See loess for more information.

peakGroupsMatrix optional matrix of (raw) retention times for the peak groups on which the align-
ment should be performed. Each column represents a sample, each row a fea-
ture/peak group. Such a matrix is for example returned by the adjustRtimePeakGroups
method.

object For adjustRtime: an XCMSnExp object containing the results from a previous
chromatographic peak detection (see findChromPeaks) and alignment analysis
(see groupChromPeaks).

For all other methods: a PeakGroupsParam object.

param A PeakGroupsParam object containing all settings for the retention time correc-
tion method.

value The value for the slot.

x a PeakGroupsParam object.

Value

The PeakGroupsParam function returns a PeakGroupsParam class instance with all of the settings
specified for retention time adjustment based on house keeping features/peak groups.

For adjustRtimePeakGroups: a matrix, rows being features, columns samples, of retention times.
The features are ordered by the median retention time across columns.

For adjustRtime: a XCMSnExp object with the results of the retention time adjustment step. These
can be accessed with the adjustedRtime method. Retention time correction does also adjust the re-
tention time of the identified chromatographic peaks (accessed via chromPeaks. Note that retention
time correction drops all previous alignment results from the result object.

Slots

__classVersion__, minFraction, extraPeaks, smooth, span, family, peakGroupsMatrix See
corresponding parameter above. __classVersion__ stores the version from the class. Slots
values should exclusively be accessed via the corresponding getter and setter methods listed
above.
Note

These methods and classes are part of the updated and modernized `xcms` user interface which will eventually replace the `group` methods. All of the settings to the alignment algorithm can be passed with a `PeakGroupsParam` object.

The matrix with the (raw) retention times of the peak groups used in the alignment is added to the `peakGroupsMatrix` slot of the `PeakGroupsParam` object that is stored into the corresponding `process history step` (see `processHistory` for how to access the process history).

`adjustRtimePeakGroups` is supposed to be called before the sample alignment, but after a correspondence (peak grouping).

This method requires that a correspondence has been performed on the data (see `groupChromPeaks`). Calling `adjustRtime` on an `XCMSnExp` object will cause all peak grouping (correspondence) results and any previous retention time adjustments to be dropped. In some instances, the `adjustRtime,XCMSnExp,PeakGroupsParam` re-adjusts adjusted retention times to ensure them being in the same order than the raw (original) retention times.

Author(s)

Colin Smith, Johannes Rainer

References


See Also

The `do_adjustRtime_peakGroups` core API function and `retcor.peakgroups` for the old user interface. `plotAdjustedRtime` for visualization of alignment results. `XCMSnExp` for the object containing the results of the alignment.

Other retention time correction methods: `adjustRtime-obiwarp, adjustRtime`

Examples

```
### Chromatographic peak detection and grouping.
##
## Below we perform first a peak detection (using the matchedFilter method) on some of the test files from the faahKO package followed by a peak grouping.
library(faahKO)
library(xcms)
fls <- dir(system.file("cdf/KO", package = "faahKO"), recursive = TRUE, full.names = TRUE)

## Reading 2 of the KO samples
raw_data <- readMSData2(fls[1:2])

## Perform the peak detection using the matchedFilter method.
mfp <- MatchedFilterParam(snthresh = 20, binSize = 1)
res <- findChromPeaks(raw_data, param = mfp)

head(chromPeaks(res))
```
## The number of peaks identified per sample:
```
table(chromPeaks(res)[, "sample"])
```

## Performing the peak grouping using the "peak density" method.
```
pp <- PeakDensityParam(sampleGroups = c(1, 1))
res <- groupChromPeaks(res, param = pp)
```

## Perform the retention time adjustment using peak groups found in both files.
```
fgp <- PeakGroupsParam(minFraction = 1)
```

## Before running the alignment we can evaluate which features (peak groups) would be used based on the specified parameters.
```
pkGrps <- adjustRtimePeakGroups(res, param = fgp)
```

## We can also plot these to evaluate if the peak groups span a large portion of the retention time range.
```
plot(x = pkGrps[, 1], y = rep(1, nrow(pkGrps)), xlim = range(rtime(res)), ylim = c(1, 2), xlab = "rt", ylab = "", yaxt = "n")
points(x = pkGrps[, 2], y = rep(2, nrow(pkGrps)))
segments(x0 = pkGrps[, 1], x1 = pkGrps[, 2], y0 = rep(1, nrow(pkGrps)), y1 = rep(2, nrow(pkGrps)))
grid()
axis(side = 2, at = c(1, 2), labels = colnames(pkGrps))
```

## Next we perform the alignment.
```
res <- adjustRtime(res, param = fgp)
```

## Any grouping information was dropped
```
hasFeatures(res)
```

## Plot the raw against the adjusted retention times.
```
plot(rtime(raw_data), rtime(res), pch = 16, cex = 0.25, col = fromFile(res))
```

## Adjusted retention times can be accessed using
```
all.equal(rtime(res), adjustedRtime(res))
```

## To get the raw, unadjusted retention times:
```
all.equal(rtime(res, adjusted = FALSE), rtime(raw_data))
```

## To extract the retention times grouped by sample/file:
```
rt <- rtime(res, bySample = TRUE)
```

### 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**

```r
object = "xcmsRaw" signature(object = "xcmsRaw")
```

**Author(s)**

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

**Examples**

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

## End(Not run)

---

**binYonX**  
Aggregate values in y for bins defined on x

**Description**

This function takes two same-sized numeric vectors `x` and `y`, bins/cuts `x` into bins (either a pre-defined number of equal-sized bins or bins of a pre-defined size) and aggregates values in `y` corresponding to `x` values falling within each bin. By default (i.e. method = "max") the maximal `y` value for the corresponding `x` values is identified. `x` is expected to be incrementally sorted and, if not, it will be internally sorted (in which case also `y` will be ordered according to the order of `x`).

**Usage**

```r
binYonX(x, y, breaks, nBins, binSize, binFromX, binToX, fromIdx = 1L,
toIdx = length(x), method = "max", baseValue, sortedX = !is.unsorted(x),
shiftByHalfBinSize = FALSE, returnIndex = FALSE)
```

**Arguments**

- `x`  
  Numeric vector to be used for binning.
- `y`  
  Numeric vector (same length than `x`) from which the maximum values for each bin should be defined. If not provided, `x` will be used.
breaks numeric(1) defining the number of desired bins.
binSize numeric(1) defining the desired bin size.
binFromX Optional numeric(1) allowing to manually specify the range of x-values to be used for binning. This will affect only the calculation of the breaks for the bins (i.e. if nBins or binSize is provided). If not provided the minimal value in the sub-set fromIdx-toIdx in input vector x will be used.
binToX Same as binFromX, but defining the maximum x-value to be used for binning.
fromIdx Integer vector defining the start position of one or multiple sub-sets of input vector x that should be used for binning.
toIdx Same as toIdx, but defining the maximum index (or indices) in x to be used for binning.
method A character string specifying the method that should be used to aggregate values in y. Allowed are "max", "min", "sum" and "mean" to identify the maximal or minimal value or to sum all values within a bin or calculate their mean value.
baseValue The base value for empty bins (i.e. bins into which either no values in x did fall, or to which only NA values in y were assigned). By default (i.e. if not specified), NA is assigned to such bins.
sortedX Whether x is sorted.
shiftByHalfBinSize Logical specifying whether the bins should be shifted by half the bin size to the left. Thus, the first bin will have its center at fromX and its lower and upper boundary are fromX - binSize/2 and fromX + binSize/2. This argument is ignored if breaks are provided.
returnIndex Logical indicating whether the index of the max (if method = "max") or min (if method = "min") value within each bin in input vector x should also be reported. For methods other than "max" or "min" this argument is ignored.

Details
The breaks defining the boundary of each bin can be either passed directly to the function with the argument breaks, or are calculated on the data based on arguments nBins or binSize along with fromIdx, toIdx and optionally binFromX and binToX. Arguments fromIdx and toIdx allow to specify subset(s) of the input vector x on which bins should be calculated. The default the full x vector is considered. Also, if not specified otherwise with arguments binFromX and binToX, the range of the bins within each of the sub-sets will be from x[fromIdx] to x[toIdx]. Arguments binFromX and binToX allow to overwrite this by manually defining the a range on which the breaks should be calculated. See examples below for more details.

Calculation of breaks: for nBins the breaks correspond to seq(min(x[fromIdx])), max(x[fromIdx]), length.out = For binSize the breaks correspond to seq(min(x[fromIdx]), max(x[toIdx])), by = binSize) with the exception that the last break value is forced to be equal to max(x[toIdx]). This ensures that all values from the specified range are covered by the breaks defining the bins. The last bin could however in some instances be slightly larger than binSize. See breaks_on_binSize and breaks_on_nBins for more details.

Value
Returns a list of length 2, the first element (named "x") contains the bin mid-points, the second element (named "y") the aggregated values from input vector y within each bin. For returnIndex = TRUE
the list contains an additional element "index" with the index of the max or min (depending on whether method = "max" or method = "min") value within each bin in input vector x.

Note

The function ensures that all values within the range used to define the breaks are considered in the binning (and assigned to a bin). This means that for all bins except the last one values in x have to be \( \geq x_{\text{lower}} \) and \( < x_{\text{upper}} \) (with \( x_{\text{lower}} \) and \( x_{\text{upper}} \) being the lower and upper boundary, respectively). For the last bin the condition is \( x \geq x_{\text{lower}} \) & \( x \leq x_{\text{upper}} \). Note also that if shiftByHalfBinSize is TRUE the range of values that is used for binning is expanded by binSize (i.e. the lower boundary will be \( fromX - \text{binSize}/2 \), the upper to \( \text{toX} + \text{binSize}/2 \)). Setting this argument to TRUE resembles the binning that is/was used in profBin function from xcms < 1.5.1.

NA handling: by default the function ignores NA values in y (thus inherently assumes na.rm = TRUE). No NA values are allowed in x.

Author(s)

Johannes Rainer

See Also

imputeLinInterpol

Examples

```
## Simple example illustrating the breaks and the binning.
## Define breaks for 5 bins:
brks <- seq(2, 12, length.out = 6)
## The first bin is then [2,4), the second [4,6) and so on.
brks
## Get the max value falling within each bin.
binYonX(x = 1:16, y = 1:16, breaks = brks)
## Thus, the largest value in x = 1:16 falling into the bin [2,4) (i.e. being
## \( \geq 2 \) and \( < 4 \)) is 3, the largest one falling into [4,6) is 5 and so on.
## Note however the function ensures that the minimal and maximal x-value
## (in this example 1 and 12) fall within a bin, i.e. 12 is considered for
## the last bin.

## Performing the binning ons sub-set of x
## x <- 1:16
## Bin X from element 4 to 10 into 5 bins.
x[4:10]
binYonX(x, x, nBins = 5L, fromIdx = 4, toIdx = 10)
## This defines breaks for 5 bins on the values from 4 to 10 and bins
## the values into these 5 bins. Alternatively, we could manually specify
## the range for the binning, i.e. the minimal and maximal value for the
## breaks:
binYonX(X, x, nBins = 5L, fromIdx = 4, toIdx = 10, binFromX = 1, binToX = 16)
## In this case the breaks for 5 bins were defined from a value 1 to 16 and
## the values 4 to 10 were binned based on these breaks.
```

## Bin values within a sub-set of x, second example

This example illustrates how the fromIdx and toIdx parameters can be used. x defines 3 times the sequence form 1 to 10, while y is the sequence from 1 to 30. In this very simple example x is supposed to represent M/Z values from 3 consecutive scans and y the intensities measured for each M/Z in each scan. We want to get the maximum intensities for M/Z value bins only for the second scan, and thus we use fromIdx = 11 and toIdx = 20. The breaks for the bins are defined with the nBins, binFromX and binToX.

```r
X <- rep(1:10, 3)
Y <- 1:30
## Bin the M/Z values in the second scan into 5 bins and get the maximum intensity for each bin. Note that we have to specify sortedX = TRUE as the x and y vectors would be sorted otherwise.
binYonX(X, Y, nBins = 5L, sortedX = TRUE, fromIdx = 11, toIdx = 20)
```

####

## Bin in overlapping sub-sets of X

In this example we define overlapping sub-sets of X and perform the binning within these.

```r
X <- 1:30
## Define the start and end indices of the sub-sets.
fIdx <- c(2, 8, 21)
tIdx <- c(10, 25, 30)
binYonX(X, nBins = 5L, fromIdx = fIdx, toIdx = tIdx)
```

```r
## The same, but pre-defining also the desired range of the bins.
binYonX(X, nBins = 5L, fromIdx = fIdx, toIdx = tIdx, binFromX = 4, binToX = 28)
## The same bins are thus used for each sub-set.
```

---

**breaks_on_binSize**

*Generate breaks for binning using a defined bin size.*

### Description

Defines breaks for binSize sized bins for values ranging from fromX to toX.

### Usage

```r
breaks_on_binSize(fromX, toX, binSize)
```

### Arguments

- **fromX**
  - numeric(1) specifying the lowest value for the bins.
- **toX**
  - numeric(1) specifying the largest value for the bins.
- **binSize**
  - numeric(1) defining the size of a bin.

### Details

This function creates breaks for bins of size binSize. The function ensures that the full data range is included in the bins, i.e. the last value (upper boundary of the last bin) is always equal toX. This however means that the size of the last bin will not always be equal to the desired bin size. See examples for more details and a comparison to R’s seq function.
**Value**

A numeric vector defining the lower and upper bounds of the bins.

**Author(s)**

Johannes Rainer

**See Also**

*binYonX* for a binning function.

Other functions to define bins: *breaks_on_nBins*

**Examples**

```r
## Define breaks with a size of 0.13 for a data range from 1 to 10:
breaks_on_binSize(1, 10, 0.13)
## The size of the last bin is however larger than 0.13:
diff(breaks_on_binSize(1, 10, 0.13))
## If we would use seq, the max value would not be included:
seq(1, 10, by = 0.13)

## In the next example we use binSize that leads to an additional last bin with
## a smaller binSize:
breaks_on_binSize(1, 10, 0.51)
## Again, the max value is included, but the size of the last bin is < 0.51.
diff(breaks_on_binSize(1, 10, 0.51))
## Using just seq would result in the following bin definition:
seq(1, 10, by = 0.51)
## Thus it defines one bin (break) less.
```

**Description**

Calculate breaks for same-sized bins for data values from `fromX` to `toX`.

**Usage**

`breaks_on_nBins(fromX, toX, nBins, shiftByHalfBinSize = FALSE)`

**Arguments**

- `fromX` numeric(1) specifying the lowest value for the bins.
- `toX` numeric(1) specifying the largest value for the bins.
- `nBins` numeric(1) defining the number of bins.
- `shiftByHalfBinSize` Logical indicating whether the bins should be shifted left by half bin size. This results centered bins, i.e. the first bin being centered at `fromX` and the last around `toX`. 
c-methods

Details

This generates bins such as a call to `seq(fromX, toX, length.out = nBins)` would. The first and second element in the result vector thus defines the lower and upper boundary for the first bin, the second and third value for the second bin and so on.

Value

A numeric vector of length `nBins + 1` defining the lower and upper bounds of the bins.

Author(s)

Johannes Rainer

See Also

`binYonX` for a binning function.

Other functions to define bins: `breaks_on_binSize`

Examples

```r
## Create breaks to bin values from 3 to 20 into 20 bins
breaks_on_nBins(3, 20, nBins = 20)
## The same call but using shiftByHalfBinSize
breaks_on_nBins(3, 20, nBins = 20, shiftByHalfBinSize = TRUE)
```

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 peaks for correcting unprecise m/z values

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

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

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

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

See Also
xcmsSet-class.
Chromatogram-class

**Description**

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

**Chromatogram**: create an instance of the Chromatogram class.

- `rtime` returns the retention times for the retention time - intensity pairs stored in the chromatogram.
- `intensity` returns the intensity for the retention time - intensity pairs stored in the chromatogram.
- `mz` get the mz (range) of the chromatogram. The function returns a numeric(2) with the lower and upper mz value.
- `precursorMz` get the mz of the precursor ion. The function returns a numeric(2) with the lower and upper mz value.
- `productMz` get the mz of the product chromatogram/ion. The function returns a numeric(2) with the lower and upper mz value.
- `aggregationFun`, `aggregationFun<-` get or set the aggregation function.
- `fromFile` returns the value from the fromFile slot.
- `length` returns the length (number of retention time - intensity pairs) of the chromatogram.
- `as.data.frame` returns the rtime and intensity values from the object as data.frame.
- `filterRt`: filters the chromatogram based on the provided retention time range.

**Usage**

```r
Chromatogram(rtime = numeric(), intensity = numeric(), mz = c(0, 0),
        filterMz = c(0, 0), precursorMz = c(NA_real_, NA_real_),
        productMz = c(NA_real_, NA_real_), fromFile = integer(),
        aggregationFun = character())
```

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

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

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

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

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

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

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

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

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

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

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

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

### Arguments

- **rtime** numeric with the retention times (length has to be equal to the length of intensity).
- **intensity** numeric with the intensity values (length has to be equal to the length of rtime).
- **mz** numeric(2) representing the mz value range (min, max) on which the chromatogram was created. This is supposed to contain the real range of mz values in contrast to the filterMz below. If not applicable use mzrange = c(0, 0).
- **filterMz** numeric(2) representing the mz value range (min, max) that was used to filter the original object on mz dimension. If not applicable use filterMz = c(0, 0).
- **precursorMz** numeric(2) for SRM/MRM transitions. Represents the mz of the precursor ion. See details for more information.
- **productMz** numeric(2) for SRM/MRM transitions. Represents the mz of the product. See details for more information.
- **fromFile** integer(1) the index of the file within the OnDiskMSnExp or XCMSnExp from which the chromatogram was extracted.
- **aggregationFun** character string specifying the function that was used to aggregate intensity values for the same retention time across the mz range. Supported are "sum" (total ion chromatogram), "max" (base peak chromatogram), "min" and "mean".
- **object** A Chromatogram object.
- **filter** For mz: whether the mz range used to filter the original object should be returned (filter = TRUE), or the mz range calculated on the real data (filter = FALSE).
- **x** For as.data.frame and length: a Chromatogram object.
- **rt** For filterRt: numeric(2) defining the lower and upper retention time for the filtering.

### Details

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

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

__classVersion__, rtime, intensity, mz, filterMz, precursorMz, productMz, fromFile, aggregationFun

See corresponding parameter above.

**Author(s)**

Johannes Rainer

**See Also**

extractChromatograms for the method to extract Chromatogram objects from XCMSnExp or OnDiskMSnExp objects.

**Examples**

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

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

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

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

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

**Description**

The `findChromPeaks` methods perform the chromatographic peak detection on LC/GC-MS data and are part of the modernized `xcms` user interface.

The implemented peak detection methods in chromatographic space are:

- centWave: chromatographic peak detection using the `centWave` method. See `centWave` for more details.
- centWave with predicted isotopes: peak detection using a two-step centWave-based approach considering also feature isotopes. See `centWaveWithPredIsoROIs` for more details.
- matchedFilter: peak detection in chromatographic space. See `matchedFilter` for more details.
- massifquant: peak detection using the Kalman filter-based method. See `massifquant` for more details.
- MSW: single-spectrum non-chromatography MS data peak detection. See `MSW` for more details.
**collect-methods**

**Author(s)**
Johannes Rainer

**See Also**
- `findPeaks` for the old peak detection methods.
- Other peak detection methods: `findChromPeaks-centWaveWithPredIsoROIs`, `findChromPeaks-centWave`, `findChromPeaks-massifquant`, `findChromPeaks-matchedFilter`, `findPeaks-MSW`

---

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

**Arguments**
- `object` (empty) `xcmsFragments-class` object
- `xs` A `xcmsSet-class` object which contains picked `ms1`-peaks from several experiments
- `compMethod` ("floor", "round", "none"): compare-method which is used to find the parent peak of a `msN`peak through comparing the MZ-values of the `ms1`peaks with the `msNParentPeaks`.
- `snthresh`, `mzgap`, `uniq`
  - these are the parameters for the getspec-peakpicker included in `xcmsRaw`.

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

```r
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**

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

**Details**

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

Additionally, it does all the heavy lifting involved in creating superimposed extracted ion chromatograms for a given number of analytes. It does so by reading the raw data files associated with the samples of interest one at a time. As it does so, it prints the name of the sample it is currently reading. Depending on the number and size of the samples, this process can take a long time.
If a base file name is provided, the report (see Value section) will be saved to a tab separated file. If EICs are generated, they will be saved as 640x480 PNG files in a newly created subdirectory. However this parameter can be changed with the commands arguments. The numbered file names correspond to the rows in the report.

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

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

**Value**

A data frame with the following columns:

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

**Methods**

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

**See Also**

`xcmsSet-class`, `mt.teststat`, `palette`
do_adjustRtime_peakGroups

Align spectrum retention times across samples using peak groups found in most samples

Description

The function performs retention time correction by assessing the retention time deviation across all samples using peak groups (features) containing chromatographic peaks present in most/all samples. The retention time deviation for these features in each sample is described by fitting either a polynomial (smooth = "loess") or a linear (smooth = "linear") model to the data points. The models are subsequently used to adjust the retention time for each spectrum in each sample.

Usage

do_adjustRtime_peakGroups(peaks, peakIndex, rtime, minFraction = 0.9, extraPeaks = 1, smooth = c("loess", "linear"), span = 0.2, family = c("gaussian", "symmetric"), peakGroupsMatrix = matrix(ncol = 0, nrow = 0))

Arguments

- **peaks**: a matrix or data.frame with the identified chromatographic peaks in the samples.
- **peakIndex**: a list of indices that provides the grouping information of the chromatographic peaks (across and within samples).
- **rtime**: a list of numeric vectors with the retention times per file/sample.
- **minFraction**: numeric(1) between 0 and 1 defining the minimum required fraction of samples in which peaks for the peak group were identified. Peak groups passing this criteria will aligned across samples and retention times of individual spectra will be adjusted based on this alignment. For minFraction = 1 the peak group has to contain peaks in all samples of the experiment.
- **extraPeaks**: numeric(1) defining the maximal number of additional peaks for all samples to be assigned to a peak group (i.e. feature) for retention time correction. For a data set with 6 samples, extraPeaks = 1 uses all peak groups with a total peak count <= 6 + 1. The total peak count is the total number of peaks being assigned to a peak group and considers also multiple peaks within a sample being assigned to the group.
- **smooth**: character defining the function to be used, to interpolate corrected retention times for all peak groups. Either "loess" or "linear".
- **span**: numeric(1) defining the degree of smoothing (if smooth = "loess"). This parameter is passed to the internal call to loess.
- **family**: character defining the method to be used for loess smoothing. Allowed values are "gaussian" and "symmetric". See loess for more information.
- **peakGroupsMatrix**: optional matrix of (raw) retention times for peak groups on which the alignment should be performed. Each column represents a sample, each row a feature/peak group. If not provided, this matrix will be determined depending on parameters minFraction and extraPeaks. If provided, minFraction and extraPeaks will be ignored.
do_findChromPeaks_centWave

Details

The alignment bases on the presence of compounds that can be found in all/most samples of an experiment. The retention times of individual spectra are then adjusted based on the alignment of the features corresponding to these *house keeping compounds*. The parameters `minFraction` and `extraPeaks` can be used to fine tune which features should be used for the alignment (i.e. which features most likely correspond to the above mentioned house keeping compounds).

Value

A list with numeric vectors with the adjusted retention times grouped by sample.

Note

The method ensures that returned adjusted retention times are increasingly ordered, just as the raw retention times.

Author(s)

Colin Smith, Johannes Rainer

References


do_findChromPeaks_centWave

*Core API function for centWave peak detection*

Description

This function performs peak density and wavelet based chromatographic peak detection for high resolution LC/MS data in centroid mode [Tautenhahn 2008].

Usage

do_findChromPeaks_centWave(mz, int, scantime, valsPerSpect, ppm = 25, peakwidth = c(20, 50), snthresh = 10, prefilter = c(3, 100), mzCenterFun = "wMean", integrate = 1, mzdiff = -0.001, fitgauss = FALSE, noise = 0, verboseColumns = FALSE, roiList = list(), firstBaselineCheck = TRUE, roiScales = NULL)

Arguments

- `mz` Numeric vector with the individual m/z values from all scans/ spectra of one file/sample.
- `int` Numeric vector with the individual intensity values from all scans/spectra of one file/sample.
- `scantime` Numeric vector of length equal to the number of spectra/scans of the data representing the retention time of each scan.
valsPerSpect numeric vector with the number of values for each spectrum.

ppm numeric(1) defining the maximal tolerated m/z deviation in consecutive scans in parts per million (ppm) for the initial ROI definition.

peakwidth numeric(2) with the expected approximate peak width in chromatographic space. Given as a range (min, max) in seconds.

snthresh numeric(1) defining the signal to noise ratio cutoff.

prefilter numeric(2): \( c(k, I) \) specifying the prefilter step for the first analysis step (ROI detection). Mass traces are only retained if they contain at least \( k \) peaks with intensity \( \geq I \).

mzCenterFun Name of the function to calculate the m/z center of the chromatographic peak. Allowed are: "wMean": intensity weighted mean of the peak’s m/z values, "mean": mean of the peak’s m/z values, "apex": use the m/z value at the peak apex, "wMeanApex3": intensity weighted mean of the m/z value at the peak apex and the m/z values left and right of it and "meanApex3": mean of the m/z value of the peak apex and the m/z values left and right of it.

integrate Integration method. For integrate = 1 peak limits are found through descent on the mexican hat filtered data, for integrate = 2 the descent is done on the real data. The latter method is more accurate but prone to noise, while the former is more robust, but less exact.

mzdiff numeric(1) representing the minimum difference in m/z dimension for peaks with overlapping retention times; can be negative to allow overlap.

fitgauss logical(1) whether or not a Gaussian should be fitted to each peak.

noise numeric(1) allowing to set a minimum intensity required for centroids to be considered in the first analysis step (centroids with intensity < noise are omitted from ROI detection).

verboseColumns logical(1) whether additional peak meta data columns should be returned.

roiList An optional list of regions-of-interest (ROI) representing detected mass traces. If ROIs are submitted the first analysis step is omitted and chromatographic peak detection is performed on the submitted ROIs. Each ROI is expected to have the following elements specified: scmin (start scan index), scmax (end scan index), mzmin (minimum m/z), mzmax (maximum m/z), length (number of scans), intensity (summed intensity). Each ROI should be represented by a list of elements or a single row data.frame.

firstBaselineCheck logical(1). If TRUE continuous data within regions of interest is checked to be above the first baseline.

roiScales Optional numeric vector with length equal to roiList defining the scale for each region of interest in roiList that should be used for the centWave-wavelets.

Details

This algorithm is most suitable for high resolution LC/(TOF,OrbiTrap,FTICR)-MS data in centroid mode. In the first phase the method identifies regions of interest (ROIs) representing mass traces that are characterized as regions with less than ppm m/z deviation in consecutive scans in the LC/MS map. These ROIs are then subsequently analyzed using continuous wavelet transform (CWT) to locate chromatographic peaks on different scales. The first analysis step is skipped, if regions of interest are passed with the roiList parameter.
Value

A matrix, each row representing an identified chromatographic peak, with columns:

- **mz**  Intensity weighted mean of m/z values of the peak across scans.
- **mzmin**  Minimum m/z of the peak.
- **mzmax**  Maximum m/z of the peak.
- **rt**  Retention time of the peak’s midpoint.
- **rtmin**  Minimum retention time of the peak.
- **rtmax**  Maximum retention time of the peak.
- **into**  Integrated (original) intensity of the peak.
- **intb**  Per-peak baseline corrected integrated peak intensity.
- **maxo**  Maximum intensity of the peak.
- **sn**  Signal to noise ratio, defined as \((\text{maxo} - \text{baseline})/\text{sd}\), sd being the standard deviation of local chromatographic noise.
- **egauss**  RMSE of Gaussian fit.

Additional columns for `verboseColumns = TRUE`:

- **mu**  Gaussian parameter mu.
- **sigma**  Gaussian parameter sigma.
- **h**  Gaussian parameter h.
- **f**  Region number of the m/z ROI where the peak was localized.
- **dppm**  m/z deviation of mass trace across scans in ppk.
- **scale**  Scale on which the peak was localized.
- **scpos**  Peak position found by wavelet analysis (scan number).
- **scmin**  Left peak limit found by wavelet analysis (scan number).
- **scmax**  Right peak limit found by wavelet analysis (scan number).

Note

The `centWave` was designed to work on centroided mode, thus it is expected that such data is presented to the function.

This function exposes core chromatographic peak detection functionality of the `centWave` method. While this function can be called directly, users will generally call the corresponding method for the data object instead.

Author(s)

Ralf Tautenhahn, Johannes Rainer

References

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

centWave for the standard user interface method.

Other core peak detection functions: do_findChromPeaks_centWaveWithPredIsoROIs, do_findChromPeaks_massifquant, do_findChromPeaks_matchedFilter, do_findPeaks_MSW

Examples

```r
## Load the test file
library(faahKO)
fs <- system.file("/quotesingle.Var/cdf/KO/ko15.CDF", package = "faahKO")
xr <- xcmsRaw(fs, profstep = 0)

## Extracting the data from the xcmsRaw for do_findChromPeaks_centWave
mzVals <- xr@env$mz
intVals <- xr@env$intensity

## Define the values per spectrum:
valsPerSpect <- diff(c(xr@scanindex, length(mzVals)))

## Calling the function. We're using a large value for noise to speed up
## the call in the example performance - in a real use case we would either
## set the value to a reasonable value or use the default value.
res <- do_findChromPeaks_centWaveWithPredIsoROIs(mz = mzVals, int = intVals, scantime = xr@scantime, valsPerSpect = valsPerSpect, noise = 10000)
head(res)
```

---

do_findChromPeaks_centWaveWithPredIsoROIs

Core API function for two-step centWave peak detection with isotopes

Description

The do_findChromPeaks_centWaveWithPredIsoROIs performs a two-step centWave based peak detection: chromatographic peaks are identified using centWave followed by a prediction of the location of the identified peaks’ isotopes in the mz-retention time space. These locations are fed as regions of interest (ROIs) to a subsequent centWave run. All non overlapping peaks from these two peak detection runs are reported as the final list of identified peaks.

The do_findChromPeaks_centWaveAddPredIsoROIs performs centWave based peak detection based in regions of interest (ROIs) representing predicted isotopes for the peaks submitted with argument peaks. The function returns a matrix with the identified peaks consisting of all input peaks and peaks representing predicted isotopes of these (if found by the centWave algorithm).

Usage

do_findChromPeaks_centWaveWithPredIsoROIs(mz, int, scantime, valsPerSpect, ppm = 25, peakwidth = c(20, 50), snthresh = 10, prefilter = c(3, 100), mzCenterFun = "wMean", integrate = 1, mzdiff = -0.001, fitgauss = FALSE, noise = 0, verboseColumns = FALSE, roiList = list(), firstBaselineCheck = TRUE, roiScales = NULL, snthreshIsoROIs = 6.25, maxCharge = 3, maxIso = 5, mzIntervalExtension = TRUE, polarity = "unknown")
do_findChromPeaks_addPredIsoROIs(mz, int, scantime, valsPerSpect, ppm = 25,
    peakwidth = c(20, 50), snthresh = 6.25, prefilter = c(3, 100),
    mzCenterFun = "wMean", integrate = 1, mzdiff = -0.001,
    fitgauss = FALSE, noise = 0, verboseColumns = FALSE, peaks. = NULL,
    maxCharge = 3, maxIso = 5, mzIntervalExtension = TRUE,
    polarity = "unknown")

Arguments

mz Numeric vector with the individual m/z values from all scans/ spectra of one file/sample.
int Numeric vector with the individual intensity values from all scans/spectra of one file/sample.
scantime Numeric vector of length equal to the number of spectra/scans of the data representing the retention time of each scan.
valsPerSpect Numeric vector with the number of values for each spectrum.
ppm numeric(1) defining the maximal tolerated m/z deviation in consecutive scans in parts per million (ppm) for the initial ROI definition.
peakwidth numeric(2) with the expected approximate peak width in chromatographic space. Given as a range (min, max) in seconds.
snthresh For do_findChromPeaks_addPredIsoROIs: numeric(1) defining the signal to noise threshold for the centWave algorithm. For do_findChromPeaks_centWaveWithPredIsoROIs: numeric(1) defining the signal to noise threshold for the initial (first) centWave run.
prefilter numeric(2): c(k, I) specifying the prefilter step for the first analysis step (ROI detection). Mass traces are only retained if they contain at least k peaks with intensity >= I.
mzCenterFun Name of the function to calculate the m/z center of the chromatographic peak. Allowed are: "wMean": intensity weighted mean of the peak's m/z values, "mean": mean of the peak's m/z values, "apex": use the m/z value at the peak apex, "wMeanApex3": intensity weighted mean of the m/z value at the peak apex and the m/z values left and right of it and "meanApex3": mean of the m/z value of the peak apex and the m/z values left and right of it.
integrate Integration method. For integrate = 1 peak limits are found through descent on the mexican hat filtered data, for integrate = 2 the descent is done on the real data. The latter method is more accurate but prone to noise, while the former is more robust, but less exact.
mzdiff numeric(1) representing the minimum difference in m/z dimension for peaks with overlapping retention times; can be negatove to allow overlap.
fitgauss logical(1) whether or not a Gaussian should be fitted to each peak.
noise numeric(1) allowing to set a minimum intensity required for centroids to be considered in the first analysis step (centroids with intensity < noise are omitted from ROI detection).
verboseColumns logical(1) whether additional peak meta data columns should be returned.
roilist An optional list of regions-of-interest (ROI) representing detected mass traces. If ROIs are submitted the first analysis step is omitted and chromatographic peak detection is performed on the submitted ROIs. Each ROI is expected to have the following elements specified: scmin (start scan index), scmax (end...
do_findChromPeaks_centWaveWithPredIsoROIs

scan index), mzmin (minimum m/z), mzmax (maximum m/z), length (number of scans), intensity (summed intensity). Each ROI should be represented by a list of elements or a single row data.frame.

firstBaselineCheck
logical(1). If TRUE continuous data within regions of interest is checked to be above the first baseline.

roiScales
Optional numeric vector with length equal to roiList defining the scale for each region of interest in roiList that should be used for the centWave-wavelets.

snthreshIsoROIs
numeric(1) defining the signal to noise ratio cutoff to be used in the second centWave run to identify peaks for predicted isotope ROIs.

maxCharge
integer(1) defining the maximal isotope charge. Isotopes will be defined for charges 1:maxCharge.

maxIso
integer(1) defining the number of isotope peaks that should be predicted for each peak identified in the first centWave run.

mzIntervalExtension
logical(1) whether the mz range for the predicted isotope ROIs should be extended to increase detection of low intensity peaks.

polarity
character(1) specifying the polarity of the data. Currently not used, but has to be "positive", "negative" or "unknown" if provided.

A matrix or xcmsPeaks object such as one returned by a call to link{do_findChromPeaks_centWave} or link{findPeaks.centWave} (both with verboseColumns = TRUE) with the peaks for which isotopes should be predicted and used for an additional peak detection using the centWave method. Required columns are: "mz", "mzmin", "mzmax", "scmin", "scmax", "scale" and "into".

Details
For more details on the centWave algorithm see centWave.

Value
A matrix, each row representing an identified chromatographic peak. All non-overlapping peaks identified in both centWave runs are reported. The matrix columns are:

mz Intensity weighted mean of m/z values of the peaks across scans.
mzmin Minimum m/z of the peaks.
mzmax Maximum m/z of the peaks.
rt Retention time of the peak's midpoint.
rtmin Minimum retention time of the peak.
rtmax Maximum retention time of the peak.
into Integrated (original) intensity of the peak.
intb Per-peak baseline corrected integrated peak intensity.
maxo Maximum intensity of the peak.
sn Signal to noise ratio, defined as (maxo - baseline)/sd, sd being the standard deviation of local chromatographic noise.
egauss RMSE of Gaussian fit.

Additional columns for verboseColumns = TRUE:
do_findChromPeaks_massifquant

mu Gaussian parameter mu.
sigma Gaussian parameter sigma.
h Gaussian parameter h.
f Region number of the m/z ROI where the peak was localized.
dppm m/z deviation of mass trace across scans in ppk.
scale Scale on which the peak was localized.
scpos Peak position found by wavelet analysis (scan number).
scmin Left peak limit found by wavelet analysis (scan number).
scmax Right peak limit found by wavelet analysis (scan number).

Author(s)

Hendrik Treutler, Johannes Rainer

See Also

Other core peak detection functions: do_findChromPeaks_centWave, do_findChromPeaks_massifquant, do_findChromPeaks_matchedFilter, do_findPeaks_MSW

do_findChromPeaks_massifquant

Core API function for massifquant peak detection

Description

Massifquant is a Kalman filter (KF)-based chromatographic peak detection for XC-MS data in centroid mode. The identified peaks can be further refined with the centWave method (see do_findChromPeaks_centWave for details on centWave) by specifying withWave = TRUE.

Usage

do_findChromPeaks_massifquant(mz, int, scantime, valsPerSpect, ppm = 10,
peakwidth = c(20, 50), snthresh = 10, prefilter = c(3, 100),
mzCenterFun = "wMean", integrate = 1, mzdiff = -0.001,
fitgauss = FALSE, noise = 0, verboseColumns = FALSE,
criticalValue = 1.125, consecMissedLimit = 2, unions = 1,
checkBack = 0, withWave = FALSE)

Arguments

mz Numeric vector with the individual m/z values from all scans/ spectra of one file/sample.
int Numeric vector with the individual intensity values from all scans/spectra of one file/sample.
scantime Numeric vector of length equal to the number of spectra/scans of the data representing the retention time of each scan.
valsPerSpect Numeric vector with the number of values for each spectrum.
ppm numeric(1) defining the maximal tolerated m/z deviation in consecutive scans in parts per million (ppm) for the initial ROI definition.

peakwidth numeric(2) with the expected approximate peak width in chromatographic space. Given as a range (min, max) in seconds.

sntthresh numeric(1) defining the signal to noise ratio cutoff.

prefilter numeric(2): c(k, I) specifying the prefilter step for the first analysis step (ROI detection). Mass traces are only retained if they contain at least k peaks with intensity >= I.

mzCenterFun Name of the function to calculate the m/z center of the chromatographic peak. Allowed are: "wMean": intensity weighted mean of the peak's m/z values, "mean": mean of the peak's m/z values, "apex": use the m/z value at the peak apex, "wMeanApex3": intensity weighted mean of the m/z value at the peak apex and the m/z values left and right of it and "meanApex3": mean of the m/z value of the peak apex and the m/z values left and right of it.

integrate Integration method. For integrate = 1 peak limits are found through descent on the mexican hat filtered data, for integrate = 2 the descent is done on the real data. The latter method is more accurate but prone to noise, while the former is more robust, but less exact.

mzdiff numeric(1) representing the minimum difference in m/z dimension for peaks with overlapping retention times; can be negative to allow overlap.

fitgauss logical(1) whether or not a Gaussian should be fitted to each peak.

noise numeric(1) allowing to set a minimum intensity required for centroids to be considered in the first analysis step (centroids with intensity < noise are omitted from ROI detection).

verboseColumns logical(1) whether additional peak meta data columns should be returned.

criticalValue numeric(1). Suggested values: (0.1-3.0). This setting helps determine the the Kalman Filter prediction margin of error. A real centroid belonging to a bonafide peak must fall within the KF prediction margin of error. Much like in the construction of a confidence interval, criticalValue loosely translates to be a multiplier of the standard error of the prediction reported by the Kalman Filter. If the peak in the XC-MS sample have a small mass deviance in ppm error, a smaller critical value might be better and vice versa.

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

unions integer(1) set to 1 if apply t-test union on segmentation; set to 0 if no t-test to be applied on chromatographically continous peaks sharing same m/z range. Explanation: With very few data points, sometimes a Kalman Filter stops tracking a peak 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 peak divided into two segments or more. With this option turned on, the program identifies segmented peaks and combines them (merges them) into one with a two sample t-test. The potential danger of this option is that some truly distinct peaks may be merged.

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

withWave logical(1) if TRUE, the peaks identified first with Massifquant are subsequently filtered with the second step of the centWave algorithm, which includes wavelet estimation.

Details

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

Value

A matrix, each row representing an identified chromatographic peak, with columns:

- mz: Intensity weighted mean of m/z values of the peaks across scans.
- mzmin: Minimum m/z of the peak.
- mzmax: Maximum m/z of the peak.
- rtmin: Minimum retention time of the peak.
- rtmax: Maximum retention time of the peak.
- rt: Retention time of the peak’s midpoint.
- into: Integrated (original) intensity of the peak.
- maxo: Maximum intensity of the peak.

If withWave is set to TRUE, the result is the same as returned by the do_findChromPeaks_centWave method.

Author(s)

Christopher Conley

References


See Also

massifquant for the standard user interface method.

Other core peak detection functions: do_findChromPeaks_centWaveWithPredIsoROIs, do_findChromPeaks_centWave, do_findChromPeaks_matchedFilter, do_findPeaks_MSW
do_findChromPeaks_matchedFilter

Core API function for matchedFilter peak detection

Description

This function identifies peaks in the chromatographic time domain as described in [Smith 2006]. The intensity values are binned by cutting the LC/MS data into slices (bins) of a mass unit (binSize m/z) wide. Within each bin the maximal intensity is selected. The peak detection is then performed in each bin by extending it based on the steps parameter to generate slices comprising bins \( \text{current} \_ \text{bin} - \text{steps} + 1 \) to \( \text{current} \_ \text{bin} + \text{steps} - 1 \). Each of these slices is then filtered with matched filtration using a second-derivative Gaussian as the model peak shape. After filtration peaks are detected using a signal-to-ration cut-off. For more details and illustrations see [Smith 2006].

Usage

do_findChromPeaks_matchedFilter(mz, int, scantime, valsPerSpect, 
  binSize = 0.1, impute = "none", baseValue, distance, fwhm = 30, 
  sigma = fwhm/2.3548, max = 5, snthresh = 10, steps = 2, mzdif = 0.8 
  - binSize * steps, index = FALSE)

Arguments

mz Numeric vector with the individual m/z values from all scans/ spectra of one file/sample.

int Numeric vector with the individual intensity values from all scans/spectra of one file/sample.

scantime Numeric vector of length equal to the number of spectra/scans of the data representing the retention time of each scan.

valsPerSpect Numeric vector with the number of values for each spectrum.

binSize numeric(1) specifying the width of the bins/slices in m/z dimension.
do_findChromPeaks_matchedFilter

impute Character string specifying the method to be used for missing value imputation. Allowed values are "none" (no linear interpolation), "lin" (linear interpolation), "linbase" (linear interpolation within a certain bin-neighborhood) and "intlin". See imputeLinInterpol for more details.

baseValue The base value to which empty elements should be set. This is only considered for method = "linbase" and corresponds to the profBinLinBase’s baselevel argument.

distance For method = "linbase": number of non-empty neighboring element of an empty element that should be considered for linear interpolation. See details section for more information.

fwhm numeric(1) specifying the full width at half maximum of matched filtration gaussian model peak. Only used to calculate the actual sigma, see below.

sigma numeric(1) specifying the standard deviation (width) of the matched filtration model peak.

max numeric(1) representing the maximum number of peaks that are expected/will be identified per slice.

snthresh numeric(1) defining the signal to noise ratio cutoff.

steps numeric(1) defining the number of bins to be merged before filtration (i.e. the number of neighboring bins that will be joined to the slice in which filtration and peak detection will be performed).

mzdiff numeric(1) representing the minimum difference in m/z dimension for peaks with overlapping retention times; can be negative to allow overlap.

index logical(1) specifying whether indices should be returned instead of values for m/z and retention times.

Details

The intensities are binned by the provided m/z values within each spectrum (scan). Binning is performed such that the bins are centered around the m/z values (i.e. the first bin includes all m/z values between min(mz) - bin_size/2 and min(mz) + bin_size/2).

For more details on binning and missing value imputation see binYonX and imputeLinInterpol methods.

Value

A matrix, each row representing an identified chromatographic peak, with columns:

- **mz** Intensity weighted mean of m/z values of the peak across scans.
- **mzmin** Minimum m/z of the peak.
- **mzmax** Maximum m/z of the peak.
- **rt** Retention time of the peak's midpoint.
- **rtmin** Minimum retention time of the peak.
- **rtmax** Maximum retention time of the peak.
- **into** Integrated (original) intensity of the peak.
- **inf** Integrated intensity of the filtered peak.
- **maxo** Maximum intensity of the peak.
- **maxf** Maximum intensity of the filtered peak.
- **i** Rank of peak in merged EIC (< max).
- **sn** Signal to noise ratio of the peak.
**Note**

This function exposes core peak detection functionality of the `matchedFilter` method. While this function can be called directly, users will generally call the corresponding method for the data object instead (e.g. the `link{findPeaks.matchedFilter}` method).

**Author(s)**

Colin A Smith, Johannes Rainer

**References**


**See Also**

`binYonX` for a binning function, `imputeLinInterpol` for the interpolation of missing values. `matchedFilter` for the standard user interface method.

Other core peak detection functions: `do_findChromPeaks_centWaveWithPredIsoROIs`, `do_findChromPeaks_centWave`, `do_findChromPeaks_massifquant`, `do_findPeaks_MSW`

**Examples**

```r
## Load the test file
library(faahKO)
fs <- system.file(’/quotesingle.Var/cdf/KO/ko15.CDF’, package = “faahKO”)
xr <- xcmsRaw(fs)

## Extracting the data from the xcmsRaw for do_findChromPeaks_centWave
mzVals <- xr@env$mz
intVals <- xr@env$intensity
## Define the values per spectrum:
valsPerSpect <- diff(c(xr@scanindex, length(mzVals)))
res <- do_findChromPeaks_matchedFilter(mz = mzVals, int = intVals,
scantime = xr@scantime, valsPerSpect = valsPerSpect)
head(res)
```

---

**do_findPeaks_MSW**

Core API function for single-spectrum non-chromatography MS data peak detection

**Description**

This function performs peak detection in mass spectrometry direct injection spectrum using a wavelet based algorithm.

**Usage**

`do_findPeaks_MSW(mz, int, snthresh = 3, verboseColumns = FALSE, ...)`
Arguments

- **mz**: Numeric vector with the individual m/z values from all scans/spectra of one file/sample.
- **int**: Numeric vector with the individual intensity values from all scans/spectra of one file/sample.
- **snthresh**: numeric(1) defining the signal to noise ratio cutoff.
- **verboseColumns**: logical(1) whether additional peak meta data columns should be returned.
- **...**: Additional parameters to be passed to the `peakDetectionCWT` function.

Details

This is a wrapper around the peak picker in Bioconductor’s MassSpecWavelet package calling `peakDetectionCWT` and `tuneInPeakInfo` functions. See the `xcmsDirect` vignette for more information.

Value

A matrix, each row representing an identified peak, with columns:

- **mz**: m/z value of the peak at the centroid position.
- **mzmin**: Minimum m/z of the peak.
- **mzmax**: Maximum m/z of the peak.
- **rt**: Always -1.
- **rtmin**: Always -1.
- **rtmax**: Always -1.
- **into**: Integrated (original) intensity of the peak.
- **maxo**: Maximum intensity of the peak.
- **intf**: Always NA.
- **maxf**: Maximum MSW-filter response of the peak.
- **sn**: Signal to noise ratio.

Author(s)

Joachim Kutzera, Steffen Neumann, Johannes Rainer

See Also

- `do_findChromPeaks_massifquant`.
- `do_findChromPeaks_matchedFilter`.
- `do_findChromPeaks_centWaveWithPredIsoROIs`.
- `do_findChromPeaks_centWave`.
- `do_findChromPeaks_MSW`.
- `MSW` for the standard user interface method.
- `peakDetectionCWT` from the MassSpecWavelet package.
do_groupChromPeaks_density

Description

The `do_groupChromPeaks_density` function performs chromatographic peak grouping based on the density (distribution) of peaks, found in different samples, along the retention time axis in slices of overlapping mz ranges.

Usage

```r
do_groupChromPeaks_density(peaks, sampleGroups, bw = 30, minFraction = 0.5, minSamples = 1, binSize = 0.25, maxFeatures = 50)
```

Arguments

- **peaks**: A matrix or data.frame with the mz values and retention times of the identified chromatographic peaks in all samples of an experiment. Required columns are "mz", "rt" and "sample". The latter should contain numeric values representing the index of the sample in which the peak was found.
- **sampleGroups**: A vector of the same length than samples defining the sample group assignments (i.e. which samples belong to which sample group).
- **bw**: numeric(1) defining the bandwidth (standard deviation of the smoothing kernel) to be used. This argument is passed to the `density` method.
- **minFraction**: numeric(1) defining the minimum fraction of samples in at least one sample group in which the peaks have to be present to be considered as a peak group (feature).
- **minSamples**: numeric(1) with the minimum number of samples in at least one sample group in which the peaks have to be detected to be considered a peak group (feature).
- **binSize**: numeric(1) defining the size of the overlapping slices in mz dimension.
- **maxFeatures**: numeric(1) with the maximum number of peak groups to be identified in a single mz slice.

Details

For overlapping slices along the mz dimension, the function calculates the density distribution of identified peaks along the retention time axis and groups peaks from the same or different samples that are close to each other. See [Smith 2006] for more details.

Value

A list with elements "featureDefinitions" and "peakIndex". "featureDefinitions" is a matrix, each row representing a (mz-rt) feature (i.e. a peak group) with columns:

- "mzmed" median of the peaks’ apex mz values.
- "mzmin" smallest mz value of all peaks’ apex within the feature.
- "mzmax" largest mz value of all peaks’ apex within the feature.
do_groupChromPeaks_density

"rtmed" the median of the peaks’ retention times.
"rtmin" the smallest retention time of the peaks in the group.
"rtmax" the largest retention time of the peaks in the group.
"npeaks" the total number of peaks assigned to the feature. Note that this number can be larger than the total number of samples, since multiple peaks from the same sample could be assigned to a feature.

"peakIndex" is a list with the indices of all peaks in a feature in the peaks input matrix.

Note

The default settings might not be appropriate for all LC/GC-MS setups, especially the bw and binSize parameter should be adjusted accordingly.

Author(s)

Colin Smith, Johannes Rainer

References


See Also

Other core peak grouping algorithms: do_groupChromPeaks_nearest, do_groupPeaks_mzClust

Examples

```r
## Load the test data set
library(faahKO)
data(faahko)

## Extract the matrix with the identified peaks from the xcmsSet:
fts <- peaks(faahko)

## Perform the peak grouping with default settings:
res <- do_groupChromPeaks_density(fts, sampleGroups = sampclass(faahko))

## The feature definitions:
head(res$featureDefinitions)

## The assignment of peaks from the input matrix to the features
head(res$peakIndex)
```
do_groupChromPeaks_nearest

Core API function for chromatic peak grouping using a nearest neighbor approach

Description

The `do_groupChromPeaks_nearest` function groups peaks across samples by creating a master peak list and assigning corresponding peaks from all samples to each peak group (i.e. feature). The method is inspired by the correspondence algorithm of mzMine [Katajamaa 2006].

Usage

```r
do_groupChromPeaks_nearest(peaks, sampleGroups, mzVsRtBalance = 10, absMz = 0.2, absRt = 15, kNN = 10)
```

Arguments

- **peaks**: A matrix or data.frame with the mz values and retention times of the identified chromatographic peaks in all samples of an experiment. Required columns are "mz", "rt" and "sample". The latter should contain numeric values representing the index of the sample in which the peak was found.
- **sampleGroups**: A vector of the same length than samples defining the sample group assignments (i.e. which samples belong to which sample group).
- **mzVsRtBalance**: numeric(1) representing the factor by which mz values are multiplied before calculating the (euclician) distance between two peaks.
- **absMz**: numeric(1) maximum tolerated distance for mz values.
- **absRt**: numeric(1) maximum tolerated distance for rt values.
- **kNN**: numeric(1) representing the number of nearest neighbors to check.

Value

A list with elements "featureDefinitions" and "peakIndex". "featureDefinitions" is a matrix, each row representing an (mz-rt) feature (i.e. peak group) with columns:

- "mzmed": median of the peaks’ apex mz values.
- "mzmin": smallest mz value of all peaks’ apex within the feature.
- "mzmax": largest mz value of all peaks’ apex within the feature.
- "rtmed": the median of the peaks’ retention times.
- "rtmin": the smallest retention time of the peaks in the feature.
- "rtmax": the largest retention time of the peaks in the feature.
- "npeaks": the total number of peaks assigned to the feature.

"peakIndex" is a list with the indices of all peaks in a feature in the peaks input matrix.

References

See Also

Other core peak grouping algorithms: do_groupChromPeaks_density, do_groupPeaks_mzClust

---

**Description**

The `do_groupPeaks_mzClust` function performs high resolution correspondence on single spectra samples.

**Usage**

```r
do_groupPeaks_mzClust(peaks, sampleGroups, ppm = 20, absMz = 0, 
                        minFraction = 0.5, minSamples = 1)
```

**Arguments**

- **peaks**: A matrix or data.frame with the mz values and retention times of the identified chromatographic peaks in all samples of an experiment. Required columns are "mz", "rt" and "sample". The latter should contain numeric values representing the index of the sample in which the peak was found.
- **sampleGroups**: A vector of the same length than samples defining the sample group assignments (i.e. which samples belong to which sample group).
- **ppm**: numeric(1) representing the relative mz error for the clustering/grouping (in parts per million).
- **absMz**: numeric(1) representing the absolute mz error for the clustering.
- **minFraction**: numeric(1) defining the minimum fraction of samples in at least one sample group in which the peaks have to be present to be considered as a peak group (feature).
- **minSamples**: numeric(1) with the minimum number of samples in at least one sample group in which the peaks have to be detected to be considered a peak group (feature).

**Value**

A list with elements "featureDefinitions" and "peakIndex". "featureDefinitions" is a matrix, each row representing an (mz-rt) feature (i.e. peak group) with columns:

- "mzmed" median of the peaks' apex mz values.
- "mzmin" smallest mz value of all peaks' apex within the feature.
- "mzmax" largest mz value of all peaks' apex within the feature.
- "rtmed" always -1.
- "rtmin" always -1.
- "rtmax" always -1.
- "npeaks" the total number of peaks assigned to the feature. Note that this number can be larger than the total number of samples, since multiple peaks from the same sample could be assigned to a group.

"peakIndex" is a list with the indices of all peaks in a peak group in the peaks input matrix.
etg

Empirically Transformed Gaussian function

Description
A general function for asymmetric chromatographic peaks.

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

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

Value
The function evaluated at times x.

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

References
extractChromatograms:

The method allows to extract chromatograms from `OnDiskMSnExp` and `XCMSnExp` objects.

### Usage

```r
## S4 method for signature 'OnDiskMSnExp'
extractChromatograms(object, rt, mz,
                     aggregationFun = "sum")

## S4 method for signature 'XCMSnExp'
extractChromatograms(object, rt, mz,
                     adjustedRtime = hasAdjustedRtime(object), aggregationFun = "sum")
```

### Arguments

- **object**: Either a `OnDiskMSnExp` or `XCMSnExp` object from which the chromatograms should be extracted.
- **rt**: `numeric(2)` defining the lower and upper boundary for the retention time range. If not specified, the full retention time range of the original data will be used. It is also possible to submit a `numeric(1)` in which case `range` is called on it to transform it to a `numeric(2)`.
- **mz**: `numeric(2)` defining the lower and upper mz value for the MS data slice. If not specified, the chromatograms will be calculated on the full mz range. It is also possible to submit a `numeric(1)` in which case `range` is called on it to transform it to a `numeric(2)`.
- **aggregationFun**: character specifying the function to be used to aggregate intensity values across the mz value range for the same retention time. Allowed values are "sum", "max", "mean" and "min".
- **adjustedRtime**: For `extractChromatograms`, `XCMSnExp`: whether the adjusted (`adjustedRtime = TRUE`) or raw retention times (`adjustedRtime = FALSE`) should be used for filtering and returned in the resulting `Chromatogram` object. Adjusted retention times are used by default if available.

### Details

Arguments `rt` and `mz` allow to specify the MS data slice from which the chromatogram should be extracted. The parameter `aggregationSum` allows to specify the function to be used to aggregate the intensities across the mz range for the same retention time. Setting `aggregationFun = "sum"` would e.g. allow to calculate the total ion chromatogram (TIC), `aggregationFun = "max"` the base peak chromatogram (BPC).
Note

Chromatogram objects extracted with extractChromatogram contain NA_real_ values if, for a given retention time, no valid measurement was available for the provided mz range.

For XCMSnExp objects, if adjusted retention times are available, the extractChromatograms method will by default report and use these (for the subsetting based on the provided parameter rt). This can be overwritten with the parameter adjustedRtime.

Author(s)

Johannes Rainer

See Also

XCMSnExp for the data object. Chromatogram for the object representing chromatographic data.

Examples

## Read some files from the faahKO package.
library(xcms)
library(faahKO)
faahko_3_files <- c(system.file("/quotesingle.Var cdf/KO/ko15.CDF", package = "faahKO"),
                  system.file("/quotesingle.Var cdf/KO/ko16.CDF", package = "faahKO"),
                  system.file("/quotesingle.Var cdf/KO/ko18.CDF", package = "faahKO"))

od <- readMSData2(faahko_3_files)

## Extract the ion chromatogram for one chromatographic peak in the data.
chrs <- extractChromatograms(od, rt = c(2700, 2900), mz = 335)

## plot the data
plot(rtime(chrs[[2]]), intensity(chrs[[2]]), type = "l", xlab = "rtime",
     ylab = "intensity", col = "000080")
for(i in c(1, 3)) {
  points(rtime(chrs[[i]]), intensity(chrs[[i]]), type = "l", col = "00000080")
}

featureValues,XCMSnExp-method

Accessing mz-rt feature data values

Description

featureValues,XCMSnExp: extract a matrix for feature values with rows representing features and columns samples. Parameter value allows to define which column from the chromPeaks matrix should be returned. Multiple chromatographic peaks from the same sample can be assigned to a feature. Parameter method allows to specify the method to be used in such cases to chose from which of the peaks the value should be returned.

Usage

## S4 method for signature 'XCMSnExp'
featureValues(object, method = c("medret", "maxint"),
              value = "index", intensity = "into", filled = TRUE)
FillChromPeaksParam-class

Arguments

- **object**: A `XCMSnExp` object providing the feature definitions.

- **method**: character specifying the method to resolve multi-peak mappings within the same sample, i.e. to define the *representative* peak for a feature in samples where more than one peak was assigned to the feature. If "medret": select the peak closest to the median retention time of the feature. If "maxint": select the peak yielding the largest signal.

- **value**: character specifying the name of the column in `chromPeaks(object)` that should be returned or "index" (the default) to return the index of the peak in the `chromPeaks(object)` matrix corresponding to the *representative* peak for the feature in the respective sample.

- **intensity**: character specifying the name of the column in the `chromPeaks(objects)` matrix containing the intensity value of the peak that should be used for the conflict resolution if `method = "maxint"`.

- **filled**: logical(1) specifying whether values for filled-in peaks should be returned or not. If `filled = FALSE`, an NA is returned in the matrix for the respective peak. See `fillChromPeaks` for details on peak filling.

Value

For featureValues: a matrix with feature values, columns representing samples, rows features. The order of the features matches the order found in the `featureDefinitions(object)` DataFrame. The rownames of the matrix are the same than those of the `featureDefinitions` DataFrame. NA is reported for features without corresponding chromatographic peak in the respective sample(s).

Note

This method is equivalent to the `groupval` for `xcmsSet` objects.

Author(s)

Johannes Rainer

See Also

- `XCMSnExp` for information on the data object. `featureDefinitions` to extract the DataFrame with the feature definitions. `hasFeatures` to evaluate whether the `XCMSnExp` provides feature definitions. `groupval` for the equivalent method on `xcmsSet` objects.

---

FillChromPeaksParam-class

Integrate areas of missing peaks
Description

The \texttt{FillChromPeaksParam} object encapsulates all settings for the signal integration for missing peaks.

- \texttt{expandMz,expandMz<-}: getter and setter for the \texttt{expandMz} slot of the object.
- \texttt{expandRt,expandRt<-}: getter and setter for the \texttt{expandRt} slot of the object.
- \texttt{ppm,ppm<-}: getter and setter for the \texttt{ppm} slot of the object.

Integrate signal in the mz-rt area of a feature (chromatographic peak group) for samples in which no chromatographic peak for this feature was identified and add it to the \texttt{chromPeaks}. Such peaks will have a value of 1 in the "is_filled" column of the \texttt{chromPeaks} matrix of the object.

Usage

\begin{verbatim}
FillChromPeaksParam(expandMz = 0, expandRt = 0, ppm = 0)
## S4 method for signature 'FillChromPeaksParam'
show(object)
## S4 method for signature 'FillChromPeaksParam'
expandMz(object)
## S4 replacement method for signature 'FillChromPeaksParam'
expandMz(object) <- value
## S4 method for signature 'FillChromPeaksParam'
expandRt(object)
## S4 replacement method for signature 'FillChromPeaksParam'
expandRt(object) <- value
## S4 method for signature 'FillChromPeaksParam'
ppm(object)
## S4 replacement method for signature 'FillChromPeaksParam'
ppm(object) <- value
## S4 method for signature 'XCMSnExp,FillChromPeaksParam'
fillChromPeaks(object, param,
              BPPARAM = bpparam())
## S4 method for signature 'XCMSnExp,missing'
fillChromPeaks(object, param,
              BPPARAM = bpparam())
\end{verbatim}

Arguments

- \texttt{expandMz} numeric(1) defining the value by which the mz width of peaks should be expanded. Each peak is expanded in mz direction by \texttt{expandMz} * their original mz width. A value of 0 means no expansion, a value of 1 grows each peak by 1 * the mz width of the peak resulting in peaks with twice their original size in mz direction (expansion by half mz width to both sides).

- \texttt{expandRt} numeric(1), same as \texttt{expandRt} but for the retention time width.
FillChromPeaksParam-class

ppm numeric(1) optionally specifying a ppm by which the mz width of the peak region should be expanded. For peaks with an mz width smaller than \( \text{mean}(c(\text{mzmin}, \text{mzmax})) \times \text{ppm} / 1e6 \), the \text{mzmin} will be replaced by \( \text{mean}(c(\text{mzmin}, \text{mzmax})) - (\text{mean}(c(\text{mzmin}, \text{mzmax})) \times \text{ppm} / 2 / 1e6) \) and \text{mzmax} by \( \text{mean}(c(\text{mzmin}, \text{mzmax})) + (\text{mean}(c(\text{mzmin}, \text{mzmax})) \times \text{ppm} / 2 / 1e6) \). This is applied before eventually expanding the mz width using the expandMz parameter.

object XCMSnExp object with identified and grouped chromatographic peaks.
value The value for the slot.
param A FillChromPeaksParam object with all settings.
BPPARAM Parallel processing settings.

Details

After correspondence (i.e. grouping of chromatographic peaks across samples) there will always be features (peak groups) that do not include peaks from every sample. The fillChromPeaks method defines intensity values for such features in the missing samples by integrating the signal in the mz-rt region of the feature. The mz-rt area is defined by the median mz and rt start and end points of the other detected chromatographic peaks for a given feature.

Adjusted retention times will be used if available.

Based on the peak finding algorithm that was used to identify the (chromatographic) peaks different internal functions are employed to guarantee that the integrated peak signal matches as much as possible the peak signal integration used during the peak detection. For peaks identified with the matchedFilter method, signal integration is performed on the profile matrix generated with the same settings used also during peak finding (using the same bin size for example). For direct injection data and peaks identified with the MSW algorithm signal is integrated only along the mz dimension. For all other methods the complete (raw) signal within the area defined by "mzmin", "mzmax", "rtmin" and "rtmax" is used.

Value

The FillChromPeaksParam function returns a FillChromPeaksParam object.
A XCMSnExp object with previously missing chromatographic peaks for features filled into its chromPeaks matrix.

Slots

- \_\_classVersion\_, expandMz, expandRt, ppm See corresponding parameter above.
- \_\_classVersion\_\_ stores the version of the class.

Note

The reported "mzmin", "mzmax", "rtmin" and "rtmax" for the filled peaks represents the actual MS area from which the signal was integrated. Note that no peak is filled in if no signal was present in a file/sample in the respective mz-rt area. These samples will still show a NA in the matrix returned by the featureValues method. This is in contrast to the fillPeaks.chrom method that returned an "into" and "maxo" of 0 for such peak areas. Growing the mz-rt area using the expandMz and expandRt might help to reduce the number of missing peak signals after filling.

Author(s)

Johannes Rainer
See Also
groupChromPeaks for methods to perform the correspondence. dropFilledChromPeaks for the method to remove filled in peaks.

Examples

## Perform the peak detection using centWave on some of the files from the
## faahKO package. Files are read using the readMSData2 from the MSnbase
## package
library(faahKO)
library(xcms)
fls <- dir(system.file("cdf/KO", package = "faahKO"), recursive = TRUE,
          full.names = TRUE)
raw_data <- readMSData2(fls[1:2])

## Create a CentWaveParam object. Note that the noise is set to 10000 to
## speed up the execution of the example - in a real use case the default
## value should be used, or it should be set to a reasonable value.
cwp <- CentWaveParam(ppm = 20, noise = 10000, snthresh = 25)
res <- findChromPeaks(raw_data, param = cwp)

## Perform the correspondence.
res <- groupChromPeaks(res, param = PeakDensityParam())

## For how many features do we lack an integrated peak signal?
sum(is.na(featureValues(res)))

## Filling missing peak data using default settings.
res <- fillChromPeaks(res)

## Get the peaks that have been filled in:
fp <- chromPeaks(res)[chromPeaks(res)$"is_filled" == 1,]
head(fp)

## Did we get a signal for all missing peaks?
sum(is.na(featureValues(res)))

## No.

## Get the process history step along with the parameters used to perform
## The peak filling:
ph <- processHistory(res, type = "Missing peak filling")[[1]]
ph

## The parameter class:
ph@param

## Drop the filled in peaks:
res <- dropFilledChromPeaks(res)

## Perform the peak filling with modified settings: allow expansion of the
## mz range by a specified ppm and expanding the mz range by mz width/2
prm <- FillChromPeaksParam(ppm = 40, expandMz = 0.5)
res <- fillChromPeaks(res, param = prm)
## Did we get a signal for all missing peaks?

```r
sum(is.na(featureValues(res)))
```

## Still the same missing peaks.

### Description

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

### Arguments

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

### Details

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

### Value

A `xcmsSet` objects with filled in peak groups.

### Methods

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

### See Also

- `xcmsSet-class`, `getPeaks`
fillPeaks.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
nSlaves: (DEPRECATED): number of slaves/cores to be used for parallel peak filling. MPI is used if installed, otherwise the snow package is employed for multicore support. If none of the two packages is available it uses the parallel package for parallel processing on multiple CPUs of the current machine. Users are advised to use the BPPARAM parameter instead.
expand.mz: Expansion factor for the m/z range used for integration.
expand.rt: Expansion factor for the retention time range used for integration.
BPPARAM: allows to define a specific parallel processing setup for the current task (see bpparam from the BiocParallel package help more information). The default uses the globally defined parallel setup.

Details

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

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

Value

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

Methods

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

See Also

xcmsSet-class, getPeaks fillPeaks
fillPeaks.MSW-methods

Integrate areas of missing peaks in FTICR-MS data

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

Arguments
object the xcmsSet object

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

Value
A xcmsSet objects with filled in peak groups.

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

Note
In contrast to the fillPeaks.chrom method the maximum intensity reported in column "maxo" is not the maximum intensity measured in the expected peak area (defined by columns "mzmin" and "mzmax"), but the largest intensity of mz value(s) closest to the "mzmed" of the feature.

See Also
xcmsSet-class, getPeaks fillPeaks

filterFile,XCMSnExp-method

XCMSnExp filtering and subsetting
Description

The methods listed on this page allow to filter and subset \texttt{XCMSnExp} objects. Most of them are inherited from the \texttt{OnDiskMSnExp} object and have been adapted for \texttt{XCMSnExp} to enable subsetting also on the preprocessing results.

\texttt{filterFile}: allows to reduce the \texttt{XCMSnExp} to data from only certain files. Identified chromatographic peaks for these files are retained while all eventually present features (peak grouping information) are dropped. By default also adjusted retention times are removed. This can be overwritten by setting \texttt{keepAdjustedRtime = TRUE}, but users should use this option with caution.

\texttt{filterMz}: filters the data set based on the provided \texttt{mz} value range. All chromatographic peaks and features (grouped peaks) falling completely within the provided \texttt{mz} value range are retained (if their minimal \texttt{mz} value is $\geq \texttt{mz[1]}$ and the maximal \texttt{mz} value $\leq \texttt{mz[2]}$). Adjusted retention times, if present, are not altered by the filtering.

\texttt{filterRt}: filters the data set based on the provided retention time range. All chromatographic peaks and features (grouped peaks) the specified retention time window are retained (i.e. if the retention time corresponding to the peak’s apex is within the specified \texttt{rt} range). If retention time correction has been performed, the method will by default filter the object by adjusted retention times. The argument \texttt{adjusted} allows to specify manually whether filtering should be performed by raw or adjusted retention times. Filtering by retention time does not drop any preprocessing results. The method returns an empty object if no spectrum or feature is within the specified retention time range.

Usage

\[
\text{## S4 method for signature 'XCMSnExp'}
\text{filterFile(object, file, keepAdjustedRtime = FALSE)}
\]

\[
\text{## S4 method for signature 'XCMSnExp'}
\text{filterMz(object, mz, msLevel., \ldots)}
\]

\[
\text{## S4 method for signature 'XCMSnExp'}
\text{filterRt(object, rt, msLevel., \ldots, adjusted = hasAdjustedRtime(object))}
\]

Arguments

\begin{itemize}
\item \textbf{object} \hspace{1cm} A \texttt{XCMSnExp} object.
\item \textbf{file} \hspace{1cm} For \texttt{filterFile}: integer defining the file index within the object to subset the object by file or character specifying the file names to subset. The indices are expected to be increasingly ordered, if not they are ordered internally.
\item \textbf{keepAdjustedRtime} \hspace{1cm} For \texttt{filterFile}: \texttt{logical(1)} defining whether the adjusted retention times should be kept, even if features are being removed (and the retention time correction being potentially performed on these features).
\item \textbf{mz} \hspace{1cm} For \texttt{filterMz}: \texttt{numeric(2)} defining the lower and upper \texttt{mz} value for the filtering.
\item \textbf{msLevel.} \hspace{1cm} For \texttt{filterMz, filterRt, numeric(1)} defining the MS level(s) to which operations should be applied or to which the object should be subsetted.
\item \textbf{rt} \hspace{1cm} For \texttt{filterRt}: \texttt{numeric(2)} defining the retention time window (lower and upper bound) for the filtering.
\end{itemize}
adjusted For filterRt: logical indicating whether the object should be filtered by original (adjusted = FALSE) or adjusted retention times (adjusted = TRUE). For spectra: whether the retention times in the individual Spectrum objects should be the adjusted or raw retention times.

Value
All methods return an XCMSnExp object.

Note
The filterFile method removes also process history steps not related to the files to which the object should be sub-setted and updates the fileIndex attribute accordingly. Also, the method does not allow arbitrary ordering of the files or re-ordering of the files within the object.

Author(s)
Johannes Rainer

See Also
XCMSnExp for base class documentation.

Examples

```r
## Load some of the files from the faahKO package.
library(faahKO)
fs <- c(system.file("cdf/KO/ko15.CDF", package = "faahKO"),
        system.file("cdf/KO/ko16.CDF", package = "faahKO"),
        system.file("cdf/KO/ko18.CDF", package = "faahKO"))
## Read the files
od <- readMSData2(fs)
## Perform peak detection on them using default matched filter settings.
mfp <- MatchedFilterParam()
xod <- findChromPeaks(od, param = mfp)
## Subset the dataset to the first and third file.
xod_sub <- filterFile(xod, file = c(1, 3))
## The number of chromatographic peaks per file for the full object
table(chromPeaks(xod)[, "sample"])
## The number of chromatographic peaks per file for the subset
table(chromPeaks(xod_sub)[, "sample"])
basename(fileNames(xod))
basename(fileNames(xod_sub))
## Filter on mz values; chromatographic peaks and features within the
## mz range are retained (as well as adjusted retention times).
xod_sub <- filterMz(xod, mz = c(300, 400))
head(chromPeaks(xod_sub))
nrow(chromPeaks(xod_sub))
nrow(chromPeaks(xod))
```
## Filter on rt values. All chromatographic peaks and features within the retention time range are retained. Filtering is performed by default on adjusted retention times, if present.

```r
oxod_sub <- filterRt(xod, rt = c(2700, 2900))
```

```
runtime(xod_sub)
head(chromPeaks(xod_sub))
range(chromPeaks(xod_sub)[, "rt"])
```

```r
nrow(chromPeaks(xod))
nrow(chromPeaks(xod_sub))
```

---

**findChromPeaks-centWave**

*Chromatographic peak detection using the centWave method*

### Description

The centWave algorithm perform peak density and wavelet based chromatographic peak detection for high resolution LC/MS data in centroid mode [Tautenhahn 2008].

The `CentWaveParam` class allows to specify all settings for a chromatographic peak detection using the centWave method. Instances should be created with the `CentWaveParam` constructor.

The `detectChromPeaks,OnDiskMSnExp,CentWaveParam` method performs chromatographic peak detection using the `centWave` algorithm on all samples from an `OnDiskMSnExp` object. `OnDiskMSnExp` objects encapsulate all experiment specific data and load the spectra data (mz and intensity values) on the fly from the original files applying also all eventual data manipulations.

```
ppm, ppm<-. getter and setter for the ppm slot of the object.
peakwidth, peakwidth<-. getter and setter for the peakwidth slot of the object.
sntthresh, snthresh<-. getter and setter for the snthresh slot of the object.
prefilter, prefilter<-. getter and setter for the prefilter slot of the object.
mzCenterFun, mzCenterFun<-. getter and setter for the mzCenterFun slot of the object.
integrate, integrate<-. getter and setter for the integrate slot of the object.
mzdiff, mzdiff<-. getter and setter for the mzdiff slot of the object.
fitgauss, fitgauss<-. getter and setter for the fitgauss slot of the object.
noise, noise<-. getter and setter for the noise slot of the object.
verboseColumns, verboseColumns<-. getter and setter for the verboseColumns slot of the object.
roiList, roiList<-. getter and setter for the roiList slot of the object.
fistBaselineCheck, firstBaselineCheck<-. getter and setter for the fistBaselineCheck slot of the object.
roiScales, roiScales<-. getter and setter for the roiScales slot of the object.
```
findChromPeaks-centWave

Usage

CentWaveParam(ppm = 25, peakwidth = c(20, 50), snthresh = 10,
    prefilter = c(3, 100), mzCenterFun = "wMean", integrate = 1L,
    mzdiff = -0.001, fitgauss = FALSE, noise = 0, verboseColumns = FALSE,
    roiList = list(), firstBaselineCheck = TRUE, roiScales = numeric())

## S4 method for signature 'OnDiskMSnExp,CentWaveParam'
findChromPeaks(object, param,
    BPPARAM = bpparam(), return.type = "XCMSnExp")

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

## S4 method for signature 'CentWaveParam'
ppm(object)

## S4 replacement method for signature 'CentWaveParam'
ppm(object) <- value

## S4 method for signature 'CentWaveParam'
peakwidth(object)

## S4 replacement method for signature 'CentWaveParam'
peakwidth(object) <- value

## S4 method for signature 'CentWaveParam'
snthresh(object)

## S4 replacement method for signature 'CentWaveParam'
snthresh(object) <- value

## S4 method for signature 'CentWaveParam'
prefilter(object)

## S4 replacement method for signature 'CentWaveParam'
prefilter(object) <- value

## S4 method for signature 'CentWaveParam'
mzCenterFun(object)

## S4 replacement method for signature 'CentWaveParam'
mzCenterFun(object) <- value

## S4 method for signature 'CentWaveParam'
integrate(f)

## S4 replacement method for signature 'CentWaveParam'
integrate(object) <- value

## S4 method for signature 'CentWaveParam'
mzdiff(object)
## S4 replacement method for signature 'CentWaveParam'
mzdifff(object) <- value

## S4 method for signature 'CentWaveParam'
fitgauss(object)

## S4 replacement method for signature 'CentWaveParam'
fitgauss(object) <- value

## S4 method for signature 'CentWaveParam'
noise(object)

## S4 replacement method for signature 'CentWaveParam'
noise(object) <- value

## S4 method for signature 'CentWaveParam'
verboseColumns(object)

## S4 replacement method for signature 'CentWaveParam'
verboseColumns(object) <- value

## S4 method for signature 'CentWaveParam'
roiList(object)

## S4 replacement method for signature 'CentWaveParam'
roiList(object) <- value

## S4 method for signature 'CentWaveParam'
firstBaselineCheck(object)

## S4 replacement method for signature 'CentWaveParam'
firstBaselineCheck(object) <- value

## S4 method for signature 'CentWaveParam'
roiScales(object)

## S4 replacement method for signature 'CentWaveParam'
roiScales(object) <- value

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ppm</td>
<td>numeric(1) defining the maximal tolerated m/z deviation in consecutive scans</td>
</tr>
<tr>
<td>peakwidth</td>
<td>numeric(2) with the expected approximate peak width in chromatographic space.</td>
</tr>
<tr>
<td>snthresh</td>
<td>numeric(1) defining the signal to noise ratio cutoff.</td>
</tr>
<tr>
<td>prefilter</td>
<td>numeric(2): (c(k, I)) specifying the prefilter step for the first analysis step (ROI detection). Mass traces are only retained if they contain at least (k) peaks with intensity (\geq I).</td>
</tr>
</tbody>
</table>
| mzCenterFun| Name of the function to calculate the m/z center of the chromatographic peak. Allowed are: "\(wMean\)": intensity weighted mean of the peak's m/z values, "\(mean\)": mean of the peak's m/z values, "\(apex\)": use the m/z value at the peak apex,
The centWave algorithm is most suitable for high resolution LC/TOF-MS data in centroid mode. In the first phase the method identifies regions of interest (ROIs) representing mass traces that are characterized as regions with less than ppm m/z deviation in consecutive scans in the LC/MS map. These ROIs are then subsequently analyzed using continuous wavelet transform (CWT) to locate chromatographic peaks on different scales. The first analysis step is skipped, if regions of interest are passed via the param parameter.

Parallel processing (one process per sample) is supported and can be configured either by the BPPARAM parameter or by globally defining the parallel processing mode using the register method from the BiocParallel package.
findChromPeaks-centWave

Value

The CentWaveParam function returns a CentWaveParam class instance with all of the settings specified for chromatographic peak detection by the centWave method.

For findChromPeaks: if return.type = "XCMSnExp" an XCMSnExp object with the results of the peak detection. If return.type = "list" a list of length equal to the number of samples with matrices specifying the identified peaks. If return.type = "xcmsSet" an xcmsSet object with the results of the peak detection.

Slots

__.classVersion__, ppm, peakwidth, snthresh, prefilter, mzCenterFun, integrate, mzdiff, fitgauss, noise, verboseColumns, roiList, firstBaselineCheck, roiScales

See corresponding parameter above. __.classVersion__ stores the version from the class.

Slots values should exclusively be accessed via the corresponding getter and setter methods listed above.

Note

These methods and classes are part of the updated and modernized xcms user interface which will eventually replace the findPeaks methods. It supports peak detection on MSnExp and OnDiskMSnExp objects (both defined in the MSnbase package). All of the settings to the centWave algorithm can be passed with a CentWaveParam object.

Author(s)

Ralf Tautenhahn, Johannes Rainer

References

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

See Also

The do_findChromPeaks_centWave core API function and findPeaks.centWave for the old user interface.

XCMSnExp for the object containing the results of the peak detection.

Other peak detection methods: chromatographic-peak-detection, findChromPeaks-centWaveWithPredIsoROIs, findChromPeaks-massifquant, findChromPeaks-matchedFilter, findPeaks-MSW

Examples

```r
## Create a CentWaveParam object. Note that the noise is set to 10000 to speed up the execution of the example - in a real use case the default value should be used, or it should be set to a reasonable value.
cwp <- CentWaveParam(ppm = 20, noise = 10000)
## Change snthresh parameter
snthresh(cwp) <- 25

cwp

## Perform the peak detection using centWave on some of the files from the faahKO package. Files are read using the readMSData2 from the MSnbase package.
```
library(faahKO)
library(xcms)
fls <- dir(system.file("cdf/KO", package = "faahKO"), recursive = TRUE,
          full.names = TRUE)
raw_data <- readMSData2(fls[1:2])

## Perform the peak detection using the settings defined above.
res <- findChromPeaks(raw_data, param = cwp)
head(chromPeaks(res))

findChromPeaks-centWaveWithPredIsoROIs

Two-step centWave peak detection considering also isotopes

Description

This method performs a two-step centWave-based chromatographic peak detection: in a first cent-
Wave run peaks are identified for which then the location of their potential isotopes in the mz-
retention time is predicted. A second centWave run is then performed on these regions of interest
(ROIs). The final list of chromatographic peaks comprises all non-overlapping peaks from both
centWave runs.

The CentWavePredIsoParam class allows to specify all settings for the two-step centWave-based
peak detection considering also predicted isotopes of peaks identified in the first centWave run.
Instances should be created with the CentWavePredIsoParam constructor. See also the documenta-
tion of the CentWaveParam for all methods and arguments this class inherits.

The findChromPeaks,OnDiskMSnExp,CentWavePredIsoParam method performs a two-step centWave-
based chromatographic peak detection on all samples from an OnDiskMSnExp object. OnDiskMSnExp
objects encapsulate all experiment specific data and load the spectra data (mz and intensity values)
on the fly from the original files applying also all eventual data manipulations.

Usage

CentWavePredIsoParam(ppm = 25, peakwidth = c(20, 50), snthresh = 10,
prefilter = c(3, 100), mzCenterFun = "wMean", integrate = 1L,
mzdiff = -0.001, fitgauss = FALSE, noise = 0, verboseColumns = FALSE,
roiList = list(), firstBaselineCheck = TRUE, roiScales = numeric(),
snthreshIsoROIs = 6.25, maxCharge = 3, maxIso = 5,
mzIntervalExtension = TRUE, polarity = "unknown")

## S4 method for signature 'OnDiskMSnExp,CentWavePredIsoParam'
findChromPeaks(object, param,
          BPPARAM = bpparam(), return.type = "XCMSnExp")
```r
## S4 method for signature 'CentWavePredIsoParam'
show(object)

## S4 method for signature 'CentWavePredIsoParam'
snthreshIsoROIs(object)

## S4 replacement method for signature 'CentWavePredIsoParam'
snthreshIsoROIs(object) <- value

## S4 method for signature 'CentWavePredIsoParam'
maxCharge(object)

## S4 replacement method for signature 'CentWavePredIsoParam'
maxCharge(object) <- value

## S4 method for signature 'CentWavePredIsoParam'
maxIso(object)

## S4 replacement method for signature 'CentWavePredIsoParam'
maxIso(object) <- value

## S4 method for signature 'CentWavePredIsoParam'
mzIntervalExtension(object)

## S4 replacement method for signature 'CentWavePredIsoParam'
mzIntervalExtension(object) <- value

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

## S4 replacement method for signature 'CentWavePredIsoParam'
polarity(object) <- value

Arguments

**ppm** numeric(1) defining the maximal tolerated m/z deviation in consecutive scans in parts per million (ppm) for the initial ROI definition.

**peakwidth** numeric(2) with the expected approximate peak width in chromatographic space. Given as a range (min, max) in seconds.

**snthresh** numeric(1) defining the signal to noise ratio cutoff.

**prefilter** numeric(2): c(k, I) specifying the prefilter step for the first analysis step (ROI detection). Mass traces are only retained if they contain at least k peaks with intensity \( \geq I \).

**mzCenterFun** Name of the function to calculate the m/z center of the chromatographic peak. Allowed are: "wMean": intensity weighted mean of the peak’s m/z values, "mean": mean of the peak’s m/z values, "apex": use the m/z value at the peak apex, "wMeanApex3": intensity weighted mean of the m/z value at the peak apex and the m/z values left and right of it and "meanApex3": mean of the m/z value of the peak apex and the m/z values left and right of it.

**integrate** Integration method. For integrate = 1 peak limits are found through descent on the mexican hat filtered data, for integrate = 2 the descent is done on
the real data. The latter method is more accurate but prone to noise, while the former is more robust, but less exact.

**mzdiff** numeric(1) representing the minimum difference in m/z dimension for peaks with overlapping retention times; can be negative to allow overlap.

**fitgauss** logical(1) whether or not a Gaussian should be fitted to each peak.

**noise** numeric(1) allowing to set a minimum intensity required for centroids to be considered in the first analysis step (centroids with intensity < noise are omitted from ROI detection).

**verboseColumns** logical(1) whether additional peak meta data columns should be returned.

**roiList** An optional list of regions-of-interest (ROI) representing detected mass traces. If ROIs are submitted the first analysis step is omitted and chromatographic peak detection is performed on the submitted ROIs. Each ROI is expected to have the following elements specified: scmin (start scan index), scmax (end scan index), mzmin (minimum m/z), mzmax (maximum m/z), length (number of scans), intensity (summed intensity). Each ROI should be represented by a list of elements or a single row data.frame.

**firstBaselineCheck** logical(1). If TRUE continuous data within regions of interest is checked to be above the first baseline.

**roiScales** Optional numeric vector with length equal to roiList defining the scale for each region of interest in roiList that should be used for the centWave-wavelets.

**snthreshIsoROIs** numeric(1) defining the signal to noise ratio cutoff to be used in the second centWave run to identify peaks for predicted isotope ROIs.

**maxCharge** integer(1) defining the maximal isotope charge. Isotopes will be defined for charges 1:maxCharge.

**maxIso** integer(1) defining the number of isotope peaks that should be predicted for each peak identified in the first centWave run.

**mzIntervalExtension** logical(1) whether the mz range for the predicted isotope ROIs should be extended to increase detection of low intensity peaks.

**polarity** character(1) specifying the polarity of the data. Currently not used, but has to be "positive", "negative" or "unknown" if provided.

**object** For findChromPeaks: an OnDiskMSnExp object containing the MS- and all other experiment-relevant data.

For all other methods: a parameter object.

**param** An CentWavePredIsoParam object with the settings for the chromatographic peak detection algorithm.

**BPPARAM** A parameter class specifying if and how parallel processing should be performed. It defaults to bpparam. See documentation of the BiocParallel for more details. If parallel processing is enabled, peak detection is performed in parallel on several of the input samples.

**return.type** Character specifying what type of object the method should return. Can be either "XCMSnExp" (default), "list" or "xcmsSet".

**value** The value for the slot.
Details

See `centWave` for details on the centWave method.

Parallel processing (one process per sample) is supported and can be configured either by the BPPARAM parameter or by globally defining the parallel processing mode using the `register` method from the BiocParallel package.

Value

The `CentWavePredIsoParam` function returns a `CentWavePredIsoParam` class instance with all of the settings specified for the two-step centWave-based peak detection considering also isotopes.

For `findChromPeaks`: if `return.type = "XCMSnExp"` an `XCMSnExp` object with the results of the peak detection. If `return.type = "list"` a list of length equal to the number of samples with matrices specifying the identified peaks. If `return.type = "xcmsSet"` an `xcmsSet` object with the results of the peak detection.

Slots

- `.__classVersion__`, ppm, peakwidth, snthresh, prefilter, mzCenterFun, integrate, mzdiff, fitgauss, noise, verboseColumns, roiList, firstBaselineCheck, roiScales, snthreshIsoROIs, maxCharge, maxIso, mzIntervalExtension, polarity

See corresponding parameter above. `.__classVersion__` stores the version from the class.

Slots values should exclusively be accessed via the corresponding getter and setter methods listed above.

Note

These methods and classes are part of the updated and modernized `xcms` user interface which will eventually replace the `findPeaks` methods. It supports chromatographic peak detection on `MSnExp` and `OnDiskMSnExp` objects (both defined in the `MSnbase` package). All of the settings to the algorithm can be passed with a `CentWavePredIsoParam` object.

Author(s)

Hendrik Treutler, Johannes Rainer

See Also

The `do_findChromPeaks_centWaveWithPredIsoROIs` core API function and `findPeaks.centWave` for the old user interface. `CentWaveParam` for the class the `CentWavePredIsoParam` extends.

`XCMSnExp` for the object containing the results of the peak detection.

Other peak detection methods: `chromatographic-peak-detection`, `findChromPeaks-centWave`, `findChromPeaks-massifquant`, `findChromPeaks-matchedFilter`, `findPeaks-MSW`.

Examples

```r
## Create a param object
p <- CentWavePredIsoParam(maxCharge = 4)
## Change snthresh parameter
snthresh(p) <- 25
p
```
Chromatographic peak detection using the massifquant method

Description

Massifquant is a Kalman filter (KF)-based chromatographic peak detection for XC-MS data in centroid mode. The identified peaks can be further refined with the centWave method (see findChromPeaks-centWave for details on centWave) by specifying withWave = TRUE.

The MassifquantParam class allows to specify all settings for a chromatographic peak detection using the massifquant method eventually in combination with the centWave algorithm. Instances should be created with the MassifquantParam constructor.

The findChromPeaks,OnDiskMSnExp,MassifquantParam method performs chromatographic peak detection using the massifquant algorithm on all samples from an OnDiskMSnExp object. OnDiskMSnExp objects encapsulate all experiment specific data and load the spectra data (mz and intensity values) on the fly from the original files applying also all eventual data manipulations.

Usage

MassifquantParam(ppm = 25, peakwidth = c(20, 50), snthresh = 10, prefilter = c(3, 100), mzCenterFun = "wMean", integrate = 1L, mzdiff = -0.001, fitgauss = FALSE, noise = 0, verboseColumns = FALSE, criticalValue = 1.125, consecMissedLimit = 2, unions = 1, checkBack = 0, withWave = FALSE)

## S4 method for signature 'OnDiskMSnExp,MassifquantParam'
findChromPeaks(object, param, BPPARAM = bpparam(), return.type = "XCMSnExp")
## S4 method for signature 'MassifquantParam'
show(object)

## S4 method for signature 'MassifquantParam'
ppm(object)

## S4 replacement method for signature 'MassifquantParam'
ppm(object) <- value

## S4 method for signature 'MassifquantParam'
peakwidth(object)

## S4 replacement method for signature 'MassifquantParam'
peakwidth(object) <- value

## S4 method for signature 'MassifquantParam'
stnthresh(object)

## S4 replacement method for signature 'MassifquantParam'
stnthresh(object) <- value

## S4 method for signature 'MassifquantParam'
prefilter(object)

## S4 replacement method for signature 'MassifquantParam'
prefilter(object) <- value

## S4 method for signature 'MassifquantParam'
mzCenterFun(object)

## S4 replacement method for signature 'MassifquantParam'
mzCenterFun(object) <- value

## S4 method for signature 'MassifquantParam'
integrate(f)

## S4 replacement method for signature 'MassifquantParam'
integrate(object) <- value

## S4 method for signature 'MassifquantParam'
mzdiff(object)

## S4 replacement method for signature 'MassifquantParam'
mzdiff(object) <- value

## S4 method for signature 'MassifquantParam'
fitgauss(object)

## S4 replacement method for signature 'MassifquantParam'
fitgauss(object) <- value

## S4 method for signature 'MassifquantParam'
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ppm</td>
<td>numeric(1) defining the maximal tolerated m/z deviation in consecutive scans</td>
</tr>
<tr>
<td></td>
<td>in parts per million (ppm) for the initial ROI definition.</td>
</tr>
<tr>
<td>peakwidth</td>
<td>numeric(2). Only the first element is used by massifquant, which specifies</td>
</tr>
<tr>
<td></td>
<td>the minimum peak length in time scans. For <code>withWave = TRUE</code> the second</td>
</tr>
<tr>
<td></td>
<td>argument represents the maximum peak length subject to being greater than</td>
</tr>
<tr>
<td></td>
<td>the minimum peak length (see also documentation of <code>do_findChromPeaks_centWave</code>).</td>
</tr>
<tr>
<td>snthresh</td>
<td>numeric(1) defining the signal to noise ratio cutoff.</td>
</tr>
<tr>
<td>prefilter</td>
<td>numeric(2). The first argument is only used if (<code>withWave = TRUE</code>); see</td>
</tr>
<tr>
<td></td>
<td><code>findChromPeaks-centWave</code> for details. The second argument specifies the min-</td>
</tr>
</tbody>
</table>
findChromPeaks-massifquant

**mzCenterFun**
Name of the function to calculate the m/z center of the chromatographic peak. Allowed are: "wMean": intensity weighted mean of the peak’s m/z values, "mean": mean of the peak’s m/z values, "apex": use the m/z value at the peak apex, "wMeanApex3": intensity weighted mean of the m/z value at the peak apex and the m/z values left and right of it and "meanApex3": mean of the m/z value of the peak apex and the m/z values left and right of it.

**integrate**
Integration method. For integrate = 1 peak limits are found through descent on the mexican hat filtered data, for integrate = 2 the descent is done on the real data. The latter method is more accurate but prone to noise, while the former is more robust, but less exact.

**mzdiff**
numeric(1) representing the minimum difference in m/z dimension for peaks with overlapping retention times; can be negative to allow overlap.

**fitgauss**
logical(1) whether or not a Gaussian should be fitted to each peak.

**noise**
numeric(1) allowing to set a minimum intensity required for centroids to be considered in the first analysis step (centroids with intensity < noise are omitted from ROI detection).

**verboseColumns**
logical(1) whether additional peak meta data columns should be returned.

**criticalValue**
numeric(1). Suggested values: (0.1-3.0). This setting helps determine the Kalman Filter prediction margin of error. A real centroid belonging to a bona fide peak must fall within the KF prediction margin of error. Much like in the construction of a confidence interval, criticalVal loosely translates to be a multiplier of the standard error of the prediction reported by the Kalman Filter. If the peak in the XC-MS sample have a small mass deviance in ppm error, a smaller critical value might be better and vice versa.

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

**unions**
integer(1) set to 1 if apply t-test union on segmentation; set to 0 if no t-test to be applied on chromatographically contiguous peaks sharing same m/z range. Explanation: With very few data points, sometimes a Kalman Filter stops tracking a peak 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 peak divided into two segments or more. With this option turned on, the program identifies segmented peaks and combines them (merges them) into one with a two sample t-test. The potential danger of this option is that some truly distinct peaks may be merged.

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

**withWave**
logical(1) if TRUE, the peaks identified first with Massifquant are subsequently filtered with the second step of the centWave algorithm, which includes wavelet estimation.
object For findChromPeaks: an OnDiskMSnExp object containing the MS- and all other experiment-relevant data. For all other methods: a parameter object.

param An MassifquantParam object containing all settings for the massifquant algorithm.

BPPARAM A parameter class specifying if and how parallel processing should be performed. It defaults to bpparam. See documentation of the BiocParallel for more details. If parallel processing is enables, peak detection is performed in parallel on several of the input samples.

return.type Character specifying what type of object the method should return. Can be either "XCMSnExp" (default), "list" or "xcmsSet".

value The value for the slot.

f For integrate: a MassifquantParam object.

Details

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

Parallel processing (one process per sample) is supported and can be configured either by the BPPARAM parameter or by globally defining the parallel processing mode using the register method from the BiocParallel package.

Value

The MassifquantParam function returns a MassifquantParam class instance with all of the settings specified for chromatographic peak detection by the massifquant method.

For findChromPeaks: if return.type = "XCMSnExp" an XCMSnExp object with the results of the peak detection. If return.type = "list" a list of length equal to the number of samples with matrices specifying the identified peaks. If return.type = "xcmsSet" an xcmsSet object with the results of the peak detection.

Slots

__classVersion__, ppm, peakwidth, snthresh, prefilter, mzCenterFun, integrate, mzdifff, fitgauss, noise, verboseColumns, criticalValue, consecMissedLimit, unions, checkBack, withWave

See corresponding parameter above. __classVersion__ stores the version from the class. Slots values should exclusively be accessed via the corresponding getter and setter methods listed above.

Note

These methods and classes are part of the updated and modernized xcms user interface which will eventually replace the findPeaks methods. It supports chromatographic peak detection on MSnExp
and \texttt{OnDiskMSnExp} objects (both defined in the \texttt{MSnbase} package). All of the settings to the \texttt{massifquant} and \texttt{centWave} algorithm can be passed with a \texttt{MassifquantParam} object.

\section*{Author(s)}
Christopher Conley, Johannes Rainer

\section*{References}

\section*{See Also}
The \texttt{do\_findChromPeaks\_massifquant} core API function and \texttt{findPeaks\_massifquant} for the old user interface. \texttt{XCMSnExp} for the object containing the results of the peak detection.

Other peak detection methods: \texttt{chromatographic\_peak\_detection}, \texttt{findChromPeaks\_centWaveWithPredIsoROIs}, \texttt{findChromPeaks\_centWave}, \texttt{findChromPeaks\_matchedFilter}, \texttt{findPeaks\_MSW}

\section*{Examples}

\begin{verbatim}
## Create a MassifquantParam object.
mqp <- MassifquantParam()
## Change sntthresh parameter
sntthresh(mqp) <- 30
mqp

## Perform the peak detection using massifquant on the files from the
## faahKO package. Files are read using the readMSData2 from the MSnbase
## package
library(faahKO)
library(MSnbase)
fls <- dir(system.file("cdf/KO", package = "faahKO"), recursive = TRUE,
           full.names = TRUE)
raw_data <- readMSData2(fls[1:2])
## Perform the peak detection using the settings defined above.
res <- findChromPeaks(raw_data, param = mqp)
head(chromPeaks(res))
\end{verbatim}

\subsubsection*{findChromPeaks-matchedFilter}

\textit{Peak detection in the chromatographic time domain}

\section*{Description}
The \textit{matchedFilter} algorithm identifies peaks in the chromatographic time domain as described in [Smith 2006]. The intensity values are binned by cutting The LC/MS data into slices (bins) of a mass unit (\texttt{binSize} m/z) wide. Within each bin the maximal intensity is selected. The chromatographic peak detection is then performed in each bin by extending it based on the \texttt{steps} parameter to
generate slices comprising bins current_bin - steps + 1 to current_bin + steps - 1. Each of these slices is then filtered with matched filtration using a second-derivative Gaussian as the model peak shape. After filtration peaks are detected using a signal-to-ratio cut-off. For more details and illustrations see [Smith 2006].

The MatchedFilterParam class allows to specify all settings for a chromatographic peak detection using the matchedFilter method. Instances should be created with the MatchedFilterParam constructor.

The findChromPeaks,OnDiskMSnExp,MatchedFilterParam method performs peak detection using the matchedFilter algorithm on all samples from an OnDiskMSnExp object. OnDiskMSnExp objects encapsulate all experiment specific data and load the spectra data (mz and intensity values) on the fly from the original files applying also all eventual data manipulations.

Usage

MatchedFilterParam(binSize = 0.1, impute = "none", baseValue = numeric(), distance = numeric(), fwhm = 30, sigma = fwhm/2.3548, max = 5, snthresh = 10, steps = 2, mzdiff = 0.8 - binSize * steps, index = FALSE)

## S4 method for signature 'OnDiskMSnExp,MatchedFilterParam'
findChromPeaks(object, param, BPPARAM = bpparam(), return.type = "XCMSnExp")

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

## S4 method for signature 'MatchedFilterParam'
binSize(object)

## S4 replacement method for signature 'MatchedFilterParam'
binSize(object) <- value

## S4 method for signature 'MatchedFilterParam'
impute(object)

## S4 replacement method for signature 'MatchedFilterParam'
impute(object) <- value
## S4 method for signature 'MatchedFilterParam'
baseValue(object)

## S4 replacement method for signature 'MatchedFilterParam'
baseValue(object) <- value

## S4 method for signature 'MatchedFilterParam'
distance(object)

## S4 replacement method for signature 'MatchedFilterParam'
distance(object) <- value

## S4 method for signature 'MatchedFilterParam'
fwhm(object)

## S4 replacement method for signature 'MatchedFilterParam'
fwhm(object) <- value

## S4 method for signature 'MatchedFilterParam'
sigma(object)

## S4 replacement method for signature 'MatchedFilterParam'
sigma(object) <- value

## S4 method for signature 'MatchedFilterParam'
max(x)

## S4 replacement method for signature 'MatchedFilterParam'
max(object) <- value

## S4 method for signature 'MatchedFilterParam'
svthresh(object)

## S4 replacement method for signature 'MatchedFilterParam'
svthresh(object) <- value

## S4 method for signature 'MatchedFilterParam'
steps(object)

## S4 replacement method for signature 'MatchedFilterParam'
steps(object) <- value

## S4 method for signature 'MatchedFilterParam'
mzdiff(object)

## S4 replacement method for signature 'MatchedFilterParam'
mzdiff(object) <- value

## S4 method for signature 'MatchedFilterParam'
index(object)

## S4 replacement method for signature 'MatchedFilterParam'
index(object) <- value

Arguments

binSize numeric(1) specifying the width of the bins/slices in m/z dimension.

impute Character string specifying the method to be used for missing value imputation. Allowed values are "none" (no linear interpolation), "lin" (linear interpolation), "linbase" (linear interpolation within a certain bin-neighborhood) and "intlin". See imputeLinInterpol for more details.

baseValue The base value to which empty elements should be set. This is only considered for method = "linbase" and corresponds to the profBinLinBase's baselevel argument.

distance For method = "linbase": number of non-empty neighboring element of an empty element that should be considered for linear interpolation. See details section for more information.

fwhm numeric(1) specifying the full width at half maximum of matched filtration gaussian model peak. Only used to calculate the actual sigma, see below.

sigma numeric(1) specifying the standard deviation (width) of the matched filtration model peak.

max numeric(1) representing the maximum number of peaks that are expected/will be identified per slice.

snthresh numeric(1) defining the signal to noise cutoff to be used in the chromatographic peak detection step.

steps numeric(1) defining the number of bins to be merged before filtration (i.e. the number of neighboring bins that will be joined to the slice in which filtration and peak detection will be performed).

mzdiff numeric(1) defining the minimum difference in m/z for peaks with overlapping retention times

index logical(1) specifying whether indicies should be returned instead of values for m/z and retention times.

object For findChromPeaks: an OnDiskMSnExp object containing the MS- and all other experiment-relevant data. For all other methods: a parameter object.

param An MatchedFilterParam object containing all settings for the matchedFilter algorithm.

BPPARAM A parameter class specifying if and how parallel processing should be performed. It defaults to bpparam. See documentation of the BiocParallel for more details. If parallel processing is enables, peak detection is performed in parallel on several of the input samples.

return.type Character specifying what type of object the method should return. Can be either "XCMSnExp" (default), "list" or "xcmsSet".

value The value for the slot.

x For max: a MatchedFilterParam object.
Details

The intensities are binned by the provided m/z values within each spectrum (scan). Binning is performed such that the bins are centered around the m/z values (i.e. the first bin includes all m/z values between $\min(mz) - \frac{\text{bin	extunderscore size}}{2}$ and $\min(mz) + \frac{\text{bin	extunderscore size}}{2}$).

For more details on binning and missing value imputation see `binYonX` and `imputeLinInterpol` methods.

Parallel processing (one process per sample) is supported and can be configured either by the `BPARAM` parameter or by globally defining the parallel processing mode using the `register` method from the BiocParallel package.

Value

The `MatchedFilterParam` function returns a `MatchedFilterParam` class instance with all of the settings specified for chromatographic detection by the `matchedFilter` method.

For `findChromPeaks`: if `return	extunderscore type = "XCMSnExp"` an `XCMSnExp` object with the results of the peak detection. If `return	extunderscore type = "list"` a list of length equal to the number of samples with matrices specifying the identified peaks. If `return	extunderscore type = "xcmsSet"` an `xcmsSet` object with the results of the peak detection.

Slots

```r
__classVersion__, binSize, impute, baseValue, distance, fwhm, sigma, max, snthresh, steps, mzdiff, index
```

See corresponding parameter above. `__classVersion__` stores the version from the class. Slots values should exclusively be accessed via the corresponding getter and setter methods listed above.

Note

These methods and classes are part of the updated and modernized `xcms` user interface which will eventually replace the `findPeaks` methods. It supports chromatographic peak detection on `MSnExp` and `OnDiskMSnExp` objects (both defined in the MSnbase package). All of the settings to the `matchedFilter` algorithm can be passed with a `MatchedFilterParam` object.

Author(s)

Colin A Smith, Johannes Rainer

References


See Also

The `do_findChromPeaks_matchedFilter` core API function and `findPeaks.matchedFilter` for the old user interface.

`XCMSnExp` for the object containing the results of the chromatographic peak detection.

Other peak detection methods: `chromatographic	extunderscore peak	extunderscore detection`, `findChromPeaks	extunderscore centWaveWithPredIsoROIs`, `findChromPeaks	extunderscore centWave`, `findChromPeaks	extunderscore massifquant`, `findPeaks	extunderscore MSW`
Examples

```r
## Create a MatchedFilterParam object
mfp <- MatchedFilterParam(binSize = 0.5)
## Change snthresh parameter
snthresh(mfp) <- 15
mfp

## Perform the peak detection using matchecFilter on the files from the
## faahKO package. Files are read using the readMSData2 from the MSnbase
## package
library(faahKO)
library(MSnbase)
fls <- dir(system.file("cdf/KO", package = "faahKO"), recursive = TRUE,
          full.names = TRUE)
raw_data <- readMSData2(fls)
## Perform the chromatographic peak detection using the settings defined
## above. Note that we are also disabling parallel processing in this
## example by registering a "SerialParam"
register(SerialParam())
res <- findChromPeaks(raw_data, param = mfp)
head(chromPeaks(res))
```

findMZ

Find fragment ions in xcmsFragment objects

Description

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

Usage

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

Arguments

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

Details

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

Value

A data frame with the following columns:

- **PrecursorMz**: The precursor m/z of the fragment
findneutral

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

See Also

findneutral,

Examples

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

## End(Not run)

---

findneutral  

Find neutral losses in xcmsFragment objects

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

Usage

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

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:

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

Author(s)

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

References


See Also

findMZ,

Examples

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

**Description**

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

**Arguments**

- `object` **xcmsRaw-class** object
- `method` Method to use for peak detection. See details.
- `...` Optional arguments to be passed along

**Details**

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

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

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

**Value**

A matrix with columns:

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

and additional columns depending on the chosen method.

**Methods**

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

**See Also**

`findPeaks.matchedFilter` `findPeaks.centWave` `findPeaks.addPredictedIsotopeFeatures`

`findPeaks.centWaveWithPredictedIsotopeROIs` `xcmsRaw-class`
**findPeaks-MSW**

**Single-spectrum non-chromatography MS data peak detection**

**Description**

Perform peak detection in mass spectrometry direct injection spectrum using a wavelet based algorithm.

The MSWParam class allows to specify all settings for a peak detection using the MSW method. Instances should be created with the MSWParam constructor.

The findChromPeaks, OnDiskMSnExp, MSWParam method performs peak detection in single-spectrum non-chromatography MS data using functionality from the MassSpecWavelet package on all samples from an OnDiskMSnExp object. OnDiskMSnExp objects encapsulate all experiment specific data and load the spectra data (mz and intensity values) on the fly from the original files applying also all eventual data manipulations.

**Usage**

```r
MSWParam(snthresh = 3, verboseColumns = FALSE, scales = c(1, seq(2, 30, 2), seq(32, 64, 4)), nearbyPeak = TRUE, peakScaleRange = 5, ampTh = 0.01, minNoiseLevel = ampTh/snthresh, ridgeLength = 24, peakThr = NULL, tuneIn = FALSE, ...)
```

```r
## S4 method for signature 'OnDiskMSnExp,MSWParam'
findChromPeaks(object, param, BPPARAM = bpparam(), return.type = "XCMSnExp")
```

```r
## S4 method for signature 'MSWParam'
show(object)
```

```r
## S4 method for signature 'MSWParam'
snthresh(object)
```

```r
## S4 replacement method for signature 'MSWParam'
```
sntthresh(object) <- value

## S4 method for signature 'MSWParam'
verboseColumns(object)

## S4 replacement method for signature 'MSWParam'
verboseColumns(object) <- value

## S4 method for signature 'MSWParam'
scales(object)

## S4 replacement method for signature 'MSWParam'
scales(object) <- value

## S4 method for signature 'MSWParam'
nearbyPeak(object)

## S4 replacement method for signature 'MSWParam'
nearbyPeak(object) <- value

## S4 method for signature 'MSWParam'
peakScaleRange(object)

## S4 replacement method for signature 'MSWParam'
peakScaleRange(object) <- value

## S4 method for signature 'MSWParam'
ampTh(object)

## S4 replacement method for signature 'MSWParam'
ampTh(object) <- value

## S4 method for signature 'MSWParam'
minNoiseLevel(object)

## S4 replacement method for signature 'MSWParam'
minNoiseLevel(object) <- value

## S4 method for signature 'MSWParam'
ridgeLength(object)

## S4 replacement method for signature 'MSWParam'
ridgeLength(object) <- value

## S4 method for signature 'MSWParam'
peakThr(object)

## S4 replacement method for signature 'MSWParam'
peakThr(object) <- value

## S4 method for signature 'MSWParam'
tuneIn(object)
## S4 replacement method for signature 'MSWParam'
```
tuneIn(object) <- value
```

## S4 method for signature 'MSWParam'
```
addParams(object)
```

## S4 replacement method for signature 'MSWParam'
```
addParams(object) <- value
```

### Arguments

- `snthresh` numeric(1) defining the signal to noise ratio cutoff.
- `verboseColumns` logical(1) whether additional peak meta data columns should be returned.
- `scales` Numeric defining the scales of the continuous wavelet transform (CWT).
- `nearbyPeak` logical(1) whether to include nearby peaks of major peaks.
- `peakScaleRange` numeric(1) defining the scale range of the peak (larger than 5 by default).
- `ampTh` numeric(1) defining the minimum required relative amplitude of the peak (ratio of the maximum of CWT coefficients).
- `minNoiseLevel` numeric(1) defining the minimum noise level used in computing the SNR.
- `ridgeLength` numeric(1) defining the minimum highest scale of the peak in 2-D CWT coefficient matrix.
- `peakThr` numeric(1) with the minimum absolute intensity (above baseline) of peaks to be picked. If provided, the smoothing function `sav.gol` function is called to estimate the local intensity.
- `tuneIn` logical(1) whether to tune in the parameter estimation of the detected peaks.
- `...` Additional parameters to be passed to the `identifyMajorPeaks` and `sav.gol` functions from the MassSpecWavelet package.
- `object` For `findChromPeaks`: an `OnDiskMSnExp` object containing the MS- and all other experiment-relevant data.
  
  For all other methods: a parameter object.
- `param` An `MSWParam` object containing all settings for the algorithm.
- `BPPARAM` A parameter class specifying if and how parallel processing should be performed. It defaults to `bpparam`. See documentation of the BiocParallel for more details. If parallel processing is enabled, peak detection is performed in parallel on several of the input samples.
- `return.type` Character specifying what type of object the method should return. Can be either "XCMSnExp" (default), "list" or "xcmsSet".
- `value` The value for the slot.

### Details

This is a wrapper for the peak picker in Bioconductor’s MassSpecWavelet package calling `peakDetectionCWT` and `tuneInPeakInfo` functions. See the `xcmsDirect` vignette for more information.

Parallel processing (one process per sample) is supported and can be configured either by the `BPPARAM` parameter or by globally defining the parallel processing mode using the `register` method from the BiocParallel package.
The `MSWParam` function returns a `MSWParam` class instance with all of the settings specified for peak detection by the `MSW` method.

For `findChromPeaks`: if `return.type` = "XCMSnExp" an `XCMSnExp` object with the results of the peak detection. If `return.type` = "list" a list of length equal to the number of samples with matrices specifying the identified peaks. If `return.type` = "xcmsSet" an `xcmsSet` object with the results of the detection.

**Slots**

`.__classVersion__`, `snthresh`, `verboseColumns`, `scales`, `nearbyPeak`, `peakScaleRange`, `ampTh`, `minNoiseLevel`, `ridgeLength`, `peakThr`, `tuneIn`, `addParams`

See corresponding parameter above. `.__classVersion__` stores the version from the class. Slots values should exclusively be accessed via the corresponding getter and setter methods listed above.

**Note**

These methods and classes are part of the updated and modernized `xcms` user interface which will eventually replace the `findPeaks` methods. It supports peak detection on `MSnExp` and `OnDiskMSnExp` objects (both defined in the `MSnbase` package). All of the settings to the algorithm can be passed with a `MSWParam` object.

**Author(s)**

Joachim Kutzera, Steffen Neumann, Johannes Rainer

**See Also**

The `do_findPeaks_MSW` core API function and `findPeaks.MSW` for the old user interface.

`XCMSnExp` for the object containing the results of the peak detection.

Other peak detection methods: chromatographic-peak-detection, `findChromPeaks-centWaveWithPredIsoROIs`, `findChromPeaks-centWave`, `findChromPeaks-massifquant`, `findChromPeaks-matchedFilter`

**Examples**

```r
## Create a MSWParam object
mp <- MSWParam()
## Change snthresh parameter
snthresh(mp) <- 15
mp

## Loading a small subset of direct injection, single spectrum files
library(msdata)
fticrf <- list.files(system.file("fticr", package = "msdata"), recursive = TRUE, full.names = TRUE)
fticr <- readMSData2(fticrf[1:2], msLevel. = 1)

## Perform the MSW peak detection on these:
p <- MSWParam(scales = c(1, 7), peakThr = 80000, ampTh = 0.005, SNR.method = "data.mean", winSize.noise = 500)
fticr <- findChromPeaks(fticr, param = p)
head(chromPeaks(fticr))
```
Description

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

Arguments

- object: xcmsSet object
- ppm: maximal tolerated m/z deviation in consecutive scans, in ppm (parts per million)
- peakwidth: Chromatographic peak width, given as range (min,max) in seconds
- prefilter: prefilter=c(k,I). Prefilter step for the first phase. Mass traces are only retained if they contain at least k peaks with intensity >= I.
- mzCenterFun: Function to calculate the m/z center of the feature: 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.
- mzdff: 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
- xcmsPeaks: peak list picked using the centWave algorithm with parameter verbose.columns set to TRUE (columns scmin and scmax needed)
- snthresh: signal to noise ratio cutoff, definition see below.
- maxcharge: max. number of the isotope charge.
- maxiso: max. number of the isotope peaks to predict for each detected feature.
- mzIntervalExtension: logical, if TRUE predicted isotope ROIs (regions of interest) are extended in the m/z dimension to increase the detection of low intensity and hence noisy peaks.
Details

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

Value

A matrix with columns:

- `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 \(\frac{\text{maxo} - \text{baseline}}{\text{sd}}\), 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
- `mu`: Gaussian parameter mu
- `sigma`: Gaussian parameter sigma
- `h`: Gaussian parameter h
- `f`: Region number of m/z ROI where the peak was localised
- `dppm`: m/z deviation of mass trace across scans in ppm
- `scale`: Scale on which the peak was localised
- `scpos`: Peak position found by wavelet analysis
- `scmin`: Left peak limit found by wavelet analysis (scan number)
- `scmax`: Right peak limit found by wavelet analysis (scan number)

Methods

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

Author(s)

Ralf Tautenhahn
References

Ralf Tautenhahn, Christoph Böttinger, and Steffen Neumann “Highly sensitive feature detection for high resolution LC/MS” BMC Bioinformatics 2008, 9:504
Hendrik Treutler and Steffen Neumann. “Prediction, detection, and validation of isotope clusters in mass spectrometry data” Submitted to Metabolites 2016, Special Issue “Bioinformatics and Data Analysis”

See Also

findPeaks.centWave findPeaks-methods xcmsRaw-class

findPeaks.centWave-methods

Feature detection for high resolution LC/MS data

Description

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

Arguments

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

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

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

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

Value
A matrix with columns:

mz  weighted (by intensity) mean of peak m/z across scans
mzmin  m/z peak minimum
mzmax  m/z peak maximum
rt  retention time of peak midpoint
rtmin  leading edge of peak retention time
rtmax  trailing edge of peak retention time
intb  integrated peak intensity
maxo  maximum peak intensity
sn  Signal/Noise ratio, defined as (maxo - baseline)/sd, where maxo is the maximum peak intensity, baseline the estimated baseline value and sd the standard deviation of local chromatographic noise.
egauss  RMSE of Gaussian fit
if verbose.columns is TRUE additionally :

mu  Gaussian parameter mu
sigma  Gaussian parameter sigma
h  Gaussian parameter h
f  Region number of m/z ROI where the peak was localised
dppm  m/z deviation of mass trace across scans in ppm
scale  Scale on which the peak was localised
scpos  Peak position found by wavelet analysis
scmin  Left peak limit found by wavelet analysis (scan number)
scmax  Right peak limit found by wavelet analysis (scan number)
**findPeaks.centWaveWithPredictedIsotopeROIs-methods**

**Methods**

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

**Author(s)**

Ralf Tautenhahn

**References**

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

**See Also**

`centWave` for the new user interface. `findPeaks-methods xcmsRaw-class`

---

**findPeaks.centWaveWithPredictedIsotopeROIs-methods**

*Feature detection with centWave and additional isotope features*

**Description**

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

**Arguments**

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

sleep: number of seconds to pause between plotting peak finding cycles

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

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

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

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

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

maxcharge: max. number of the isotope charge.

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

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

Details

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

Value

A matrix with columns:

mz: weighted (by intensity) mean of peak m/z across scans
mzmin: m/z peak minimum
mzmax: m/z peak maximum
rt: retention time of peak midpoint
rtmin: leading edge of peak retention time
rtmax: trailing edge of peak retention time
into: integrated peak intensity
intb: baseline corrected integrated peak intensity
maxo: maximum peak intensity
findPeaks.massifquant-methods

<table>
<thead>
<tr>
<th>sn</th>
<th>Signal/Noise ratio, defined as (maxo - baseline)/sd, where maxo is the maximum peak intensity, baseline the estimated baseline value and sd the standard deviation of local chromatographic noise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>sgauss</td>
<td>RMSE of Gaussian fit</td>
</tr>
<tr>
<td>mu</td>
<td>Gaussian parameter ( \mu )</td>
</tr>
<tr>
<td>sigma</td>
<td>Gaussian parameter ( \sigma )</td>
</tr>
<tr>
<td>h</td>
<td>Gaussian parameter ( h )</td>
</tr>
<tr>
<td>f</td>
<td>Region number of ( m/z ) ROI where the peak was localised</td>
</tr>
<tr>
<td>dpmm</td>
<td>( m/z ) deviation of mass trace across scans in ppm</td>
</tr>
<tr>
<td>scale</td>
<td>Scale on which the peak was localised</td>
</tr>
<tr>
<td>scpos</td>
<td>Peak position found by wavelet analysis</td>
</tr>
<tr>
<td>scmin</td>
<td>Left peak limit found by wavelet analysis (scan number)</td>
</tr>
<tr>
<td>scmax</td>
<td>Right peak limit found by wavelet analysis (scan number)</td>
</tr>
</tbody>
</table>

Methods

```r
object = "xcmsRaw" findPeaks.centWaveWithPredictedIsotopeROIs(object, ppm=25, peakwidth=c(20, 50), ...) firstBaselineCheck=TRUE, roiScales=NULL, snthreshIsoROIs=6.25, maxcharge=3, maxiso=5, mzIntervalExtension=TRUE)
```

Author(s)

Ralf Tautenhahn

References


See Also

do_findChromPeaks_centWaveWithPredIsoROIs for the corresponding core API function. findPeaks.addPredictedIsotopeFeatures findPeaks.centWave findPeaks-methods xcmsRaw-class

---

findPeaks.massifquant-methods

*Feature detection for XC-MS data.*

Description

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

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

- **object**
  An xcmsRaw object.

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

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

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

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

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

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

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

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

Details

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

Value

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

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mz</td>
<td>weighted m/z mean (weighted by intensity) of the feature</td>
</tr>
<tr>
<td>mzmin</td>
<td>m/z lower boundary of the feature</td>
</tr>
<tr>
<td>mzmax</td>
<td>m/z upper boundary of the feature</td>
</tr>
<tr>
<td>rtmin</td>
<td>starting scan time of the feature</td>
</tr>
<tr>
<td>rtmax</td>
<td>starting scan time of the feature</td>
</tr>
<tr>
<td>into</td>
<td>the raw quantitation (area under the curve) of the feature.</td>
</tr>
<tr>
<td>area</td>
<td>feature area that is not normalized by the scan rate.</td>
</tr>
</tbody>
</table>

Methods

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

Author(s)

Christopher Conley

References


See Also

`centWave` for the new user interface. `findPeaks-methods xcmsSet xcmsRaw xcmsRaw-class`

Examples

```r
library(faahKO)
library(xcms)
#load all the wild type and Knock out samples
cdfpath <- system.file("cdf", package = "faahKO")
# Subset to only the first 2 files.
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)[1:2]

## Run the massifquant analysis. Setting the noise level to 10000 to speed up
```
## execution of the examples - in a real use case it should be set to a reasonable
## value.
xset <- xcmsSet(cdffiles, method = "massifquant",
    consecMissedLimit = 1,
    snthresh = 10,
    criticalValue = 1.73,
    ppm = 10,
    peakwidth= c(30, 60),
    prefilter= c(1,3000),
    noise = 10000,
    withWave = 0)

findPeaks.matchedFilter, xcmsRaw-method

Peak detection in the chromatographic time domain

Description

Find peaks in the chromatographic time domain of the profile matrix. For more details see do_findChromPeaks_matchedFilter.

Usage

```r
## S4 method for signature '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,
    scanrange = numeric())
```

Arguments

- `object`: The `xcmsRaw` object on which peak detection should be performed.
- `fwhm`: numeric(1) specifying the full width at half maximum of matched filtration gaussian model peak. Only used to calculate the actual sigma, see below.
- `sigma`: numeric(1) specifying the standard deviation (width) of the matched filtration model peak.
- `max`: numeric(1) representing the maximum number of peaks that are expected/will be identified per slice.
- `snthresh`: numeric(1) defining the signal to noise cutoff to be used in the chromatographic peak detection step.
- `step`: numeric(1) specifying the width of the bins/slices in m/z dimension.
- `steps`: numeric(1) defining the number of bins to be merged before filtration (i.e. the number of neighboring bins that will be joined to the slice in which filtration and peak detection will be performed).
- `mzdiff`: numeric(1) defining the minimum difference in m/z for peaks with overlapping retention times
- `index`: logical(1) specifying whether indicies should be returned instead of values for m/z and retention times.
- `sleep`: (DEFUNCT). This parameter is no longer functional, as it would cause problems in parallel processing mode.
- `scanrange`: Numeric vector defining the range of scans to which the original object should be sub-setted before peak detection.
**Value**

A matrix, each row representing an identified chromatographic peak, with columns:

- **mz**: Intensity weighted mean of m/z values of the peak across scans.
- **mzmin**: Minimum m/z of the peak.
- **mzmax**: Maximum m/z of the peak.
- **rt**: Retention time of the peak’s midpoint.
- **rtmin**: Minimum retention time of the peak.
- **rtmax**: Maximum retention time of the peak.
- **into**: Integrated (original) intensity of the peak.
- **intf**: Integrated intensity of the filtered peak.
- **maxo**: Maximum intensity of the peak.
- **maxf**: Maximum intensity of the filtered peak.
- **i**: Rank of peak in merged EIC ($\leq$ max).
- **sn**: Signal to noise ratio of the peak.

**Author(s)**

Colin A. Smith

**References**


**See Also**

matchedFilter for the new user interface. xcmsRaw, do_findChromPeaks_matchedFilter for the core function performing the peak detection.

---

**Description**

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

**Arguments**

- **object**: xcmsRaw object
Details

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

Since xcmsFragments attaches all MSSⁿ 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,xcmsRaw-method

Peak detection for single-spectrum non-chromatography MS data

Description
This method performs peak detection in mass spectrometry direct injection spectrum using a wavelet based algorithm.

Usage
```r
## S4 method for signature 'xcmsRaw'
findPeaks.MSW(object, snthresh = 3,
    verbose.columns = FALSE, ...)
```

Arguments
- `object` The `xcmsRaw` object on which peak detection should be performed.
- `snthresh` numeric(1) defining the signal to noise ratio cutoff.
- `verbose.columns` Logical whether additional peak meta data columns should be returned.
- `...` Additional parameters to be passed to the `identifyMajorPeaks` and `sav.gol` functions from the MassSpecWavelet package.

Details
This is a wrapper around the peak picker in Bioconductor’s MassSpecWavelet package calling `peakDetectionCWT` and `tuneInPeakInfo` functions.

Value
A matrix, each row representing an identified peak, with columns:
- `mz` m/z value of the peak at the centroid position.
- `mzmin` Minimum m/z of the peak.
- `mzmax` Maximum m/z of the peak.
- `rt` Always -1.
- `rtmin` Always -1.
- `rtmax` Always -1.
- `into` Integrated (original) intensity of the peak.
- `maxo` Maximum intensity of the peak.
- `intf` Always NA.
- `maxf` Maximum MSW-filter response of the peak.
- `sn` Signal to noise ratio.

Author(s)
Joachim Kutzera, Steffen Neumann, Johannes Rainer
**GenericParam-class**

**Description**

The `GenericParam` class allows to store generic parameter information such as the name of the function that was/has to be called (slot `fun`) and its arguments (slot `args`). This object is used to track the process history of the data processings of an `XCMSnExp` object. This is in contrast to e.g. the `CentWaveParam` object that is passed to the actual processing method.

**Usage**

```r
GenericParam(fun = character(), args = list())
```

```r
## S4 method for signature 'GenericParam'
show(object)
```

**Arguments**

- `fun` character representing the name of the function.
- `args` list (ideally named) with the arguments to the function.
- `object` `GenericParam` object.

**Value**

The `GenericParam` function returns a `GenericParam` object.

**Slots**

- `fun` character specifying the function name.
- `args` list (ideally named) with the arguments to the function.
- `.__classVersion__` the version of the class.

**Author(s)**

Johannes Rainer

**See Also**

`processHistory` for how to access the process history of an `XCMSnExp` object.

**Examples**

```r
prm <- GenericParam(fun = "mean")
prm <- GenericParam(fun = "mean", args = list(na.rm = TRUE))
```
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.

Note that this method will always return profile, not raw data (with profile data being the binned data along M/Z). See details for further information.

Arguments

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

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

step step (bin) size to use for profile generation. Note that a value of step = 0 is not supported.
groupidx either character vector with names or integer vector with indices of peak groups for which to get EICs

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

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

Details

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

Value

For xcmsSet and xcmsRaw objects, an xcmsEIC object.

Methods

object = "xcmsRaw" getEIC(object, mzrange, rtrange = NULL, step = 0.1)
object = "xcmsSet" getEIC(object, mzrange, rtrange = 200, groupidx, sampleidx = samplenames(object),

See Also

xcmsRaw-class, xcmsSet-class, xcmsEIC-class, rawEIC
**getPeaks-methods**  
*Get peak intensities for specified regions*

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

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

**Value**
A matrix with columns:
- `i`: rank of peak identified in merged EIC (< max), always NA
- `mz`: weighted (by intensity) mean of peak m/z across scans
- `mzmin`: m/z of minimum step
- `mzmax`: m/z of maximum step
- `ret`: retention time of peak midpoint
- `retmin`: leading edge of peak retention time
- `retmax`: trailing edge of peak retention time
- `into`: integrated area of original (raw) peak
- `intf`: integrated area of filtered peak, always NA
- `maxo`: maximum intensity of original (raw) peak
- `maxf`: maximum intensity of filtered peak, always NA

**Methods**
- `object = "xcmsRaw"`  
`getPeaks(object, peakrange, step = 0.1)`

**See Also**
`xcmsRaw-class`
getScan-methods

Get \( \text{m/z} \) 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** \( \text{the xcmsRaw object} \)
- **scan** \( \text{integer index of scan. if negative, the index numbered from the end} \)
- **mzrange** \( \text{limit data points returned to those between in the range, range(mzrange)} \)

**Value**

A matrix with two columns:

- \( \text{mz} \) \( \text{m/z values} \)
- \( \text{intensity} \) \( \text{intensity values} \)

**Methods**

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

**See Also**

`xcmsRaw-class`, `getSpec`

getSpec-methods

Get average \( \text{m/z} \) and intensity values for multiple mass scans

**Description**

Return full-resolution averaged data from multiple mass scans.

**Arguments**

- **object** \( \text{the xcmsRaw object} \)
- **...** \( \text{arguments passed to profRange used to sepecified the spectral segments of interest for averaging} \)

**Details**

Based on the mass points from the spectra selected, a master unique list of masses is generated. Every spectra is interpolated at those masses and then averaged.
Value
A matrix with two columns:

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

Methods

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

See Also

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

---

### Description

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

### Arguments

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

### Value

A single xcmsRaw object or a list of xcmsRaw objects.

### Methods

```r
object = "xcmsSet" getXcmsRaw(object, sampleidx=1, profmethod=profinfo(object)$method, profstep=...)
```

### Author(s)

Johannes Rainer, <johannes.rainer@eurac.edu>

### See Also

- `xcmsRaw-class`
Description

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

Arguments

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

Details

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

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

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

Value

An xcmsSet object with peak group assignments and statistics.

Methods

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

See Also

group.density group.mzClust group.nearest xcmsSet-class.
**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object</code></td>
<td>the <code>xcmsSet</code> object</td>
</tr>
<tr>
<td><code>minfrac</code></td>
<td>minimum fraction of samples necessary in at least one of the sample groups for it to be a valid group</td>
</tr>
<tr>
<td><code>minsamp</code></td>
<td>minimum number of samples necessary in at least one of the sample groups for it to be a valid group</td>
</tr>
<tr>
<td><code>bw</code></td>
<td>bandwidth (standard deviation or half width at half maximum) of gaussian smoothing kernel to apply to the peak density chromatogram</td>
</tr>
<tr>
<td><code>mzwid</code></td>
<td>width of overlapping m/z slices to use for creating peak density chromatograms and grouping peaks across samples</td>
</tr>
<tr>
<td><code>max</code></td>
<td>maximum number of groups to identify in a single m/z slice</td>
</tr>
<tr>
<td><code>sleep</code></td>
<td>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.</td>
</tr>
</tbody>
</table>

**Value**

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

**Methods**

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

**See Also**

- `do_groupChromPeaks_density` for the core API function performing the analysis. `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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object</code></td>
<td>a <code>xcmsSet</code> with peaks</td>
</tr>
<tr>
<td><code>mzppm</code></td>
<td>the relative error used for clustering/grouping in ppm (parts per million)</td>
</tr>
<tr>
<td><code>mzabs</code></td>
<td>the absolute error used for clustering/grouping</td>
</tr>
<tr>
<td><code>minsamp</code></td>
<td>set the minimum number of samples in one bin</td>
</tr>
<tr>
<td><code>minfrac</code></td>
<td>set the minimum fraction of each class in one bin</td>
</tr>
</tbody>
</table>

**Value**

Returns a `xcmsSet` with slots `groups` and `groupindex` set.
Methods

```r
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

See Also

`xcmsSet-class`

Examples

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

---

### group.nearest

**Group peaks from different samples together**

#### Description


Currently, there is no equivalent to `minfrac` or `minsamp`.

#### Arguments

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

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

Methods

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

See Also

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

Examples

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

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

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

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

gxset.post<-gxset ## copy the xcmsSet object
gxset.post@groupidx<-gxset@groupidx[ix]
gxset.post@groups<-gxset@groups[ix,]
nrow(gxset.post@groups) == 465 ## this is the number of features after minFrac

## End(Not run)
```
groupChromPeaks

Correspondence: Chromatographic peak grouping methods.

Description

The `groupChromPeaks` method(s) perform the correspondence, i.e., the grouping of chromatographic peaks within and between samples. These methods are part of the modernized `xcms` user interface. The resulting peak groups are referred to as (mz-rt) features and can be accessed via the `featureDefinitions` method on the result object.

The implemented peak grouping methods are:

- **density** peak grouping based on time dimension peak densities. See `groupChromPeaks-density` for more details.
- **mzClust** high resolution peak grouping for single spectra (direct infusion) MS data. See `groupChromPeaks-mzClust` for more details.
- **nearest** chromatographic peak grouping based on their proximity in the mz-rt space. See `groupChromPeaks-nearest` for more details.

Author(s)

Johannes Rainer

See Also

`group` for the old peak grouping methods. `featureDefinitions` and `featureValues`, `XCMSnExp-method` for methods to access peak grouping results.

Other peak grouping methods: `groupChromPeaks-density, groupChromPeaks-mzClust, groupChromPeaks-nearest`

---

**groupChromPeaks-density**

*Peak grouping based on time dimension peak densities*

Description

This method performs correspondence (chromatographic peak grouping) based on the density (distribution) of identified peaks along the retention time axis within slices of overlapping mz ranges. All peaks (from the same or from different samples) being close on the retention time axis are grouped into a feature (*peak group*).

The PeakDensityParam class allows to specify all settings for the peak grouping based on peak densities along the time dimension. Instances should be created with the PeakDensityParam constructor.

- `sampleGroups, sampleGroups<-`: getter and setter for the `sampleGroups` slot of the object.
- `bw, bw<-`: getter and setter for the `bw` slot of the object.
- `minFraction, minFraction<-`: getter and setter for the `minFraction` slot of the object.
- `minSamples, minSamples<-`: getter and setter for the `minSamples` slot of the object.
- `binSize, binSize<-`: getter and setter for the `binSize` slot of the object.
maxFeatures,maxFeatures<-: getter and setter for the maxFeatures slot of the object.

groupChromPeaks,XCMSnExp,PeakDensityParam: performs correspondence (peak grouping within
and across samples) within in mz dimension overlapping slices of MS data based on the density dis-
tribution of the identified chromatographic peaks in the slice along the time axis.

Usage

PeakDensityParam(sampleGroups = numeric(), bw = 30, minFraction = 0.5,
minSamples = 1, binSize = 0.25, maxFeatures = 50)

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

## S4 method for signature 'PeakDensityParam'
sampleGroups(object)

## S4 replacement method for signature 'PeakDensityParam'
sampleGroups(object) <- value

## S4 method for signature 'PeakDensityParam'
bw(object)

## S4 replacement method for signature 'PeakDensityParam'
bw(object) <- value

## S4 method for signature 'PeakDensityParam'
minFraction(object)

## S4 replacement method for signature 'PeakDensityParam'
minFraction(object) <- value

## S4 method for signature 'PeakDensityParam'
minSamples(object)

## S4 replacement method for signature 'PeakDensityParam'
minSamples(object) <- value

## S4 method for signature 'PeakDensityParam'
binSize(object)

## S4 replacement method for signature 'PeakDensityParam'
binSize(object) <- value

## S4 method for signature 'PeakDensityParam'
maxFeatures(object)

## S4 replacement method for signature 'PeakDensityParam'
maxFeatures(object) <- value

## S4 method for signature 'XCMSnExp,PeakDensityParam'
groupChromPeaks(object, param)
Arguments

- **sampleGroups**: A vector of the same length than samples defining the sample group assignments (i.e. which samples belong to which sample group).
- **bw**: numeric(1) defining the bandwidth (standard deviation of the smoothing kernel) to be used. This argument is passed to the `density` method.
- **minFraction**: numeric(1) defining the minimum fraction of samples in at least one sample group in which the peaks have to be present to be considered as a peak group (feature).
- **minSamples**: numeric(1) with the minimum number of samples in at least one sample group in which the peaks have to be detected to be considered a peak group (feature).
- **binSize**: numeric(1) defining the size of the overlapping slices in mz dimension.
- **maxFeatures**: numeric(1) with the maximum number of peak groups to be identified in a single mz slice.
- **object**: For `groupChromPeaks`: an `XCMSnExp` object containing the results from a previous peak detection analysis (see `findChromPeaks`). For all other methods: a `PeakDensityParam` object.

Value

The `PeakDensityParam` function returns a `PeakDensityParam` class instance with all of the settings specified for chromatographic peak alignment based on peak densities.

For `groupChromPeaks`: a `XCMSnExp` object with the results of the correspondence analysis. The definition of the resulting mz-rt features can be accessed with the `featureDefinitions` method.

Slots

```R
.__classVersion__, sampleGroups, bw, minFraction, minSamples, binSize, maxFeatures
```

See corresponding parameter above. `.__classVersion__` stores the version from the class. Slots values should exclusively be accessed via the corresponding getter and setter methods listed above.

Note

These methods and classes are part of the updated and modernized `xcms` user interface which will eventually replace the `group` methods. All of the settings to the algorithm can be passed with a `PeakDensityParam` object.

Calling `groupChromPeaks` on an `XCMSnExp` object will cause all eventually present previous correspondence results to be dropped.

Author(s)

Colin Smith, Johannes Rainer

References

See Also

The `do_groupChromPeaks_density` core API function and `group.density` for the old user interface.

`featureDefinitions` and `featureValues,XCMSnExp-method` for methods to access the features (i.e. the peak grouping results).

`XCMSnExp` for the object containing the results of the correspondence.

Other peak grouping methods: `groupChromPeaks-mzClust,groupChromPeaks-nearest,groupChromPeaks`

Examples

```r
## Create a PeakDensityParam object
p <- PeakDensityParam(binSize = 0.05)
## Change the minSamples slot
minSamples(p) <- 3
p

# Chromatographic peak detection and grouping.

## Below we perform first a peak detection (using the matchedFilter method) on some of the test files from the faahKO package followed by a peak grouping using the density method.
library(faahKO)
library(MSnbase)
fls <- dir(system.file("cdf/KO", package = "faahKO"), recursive = TRUE, full.names = TRUE)

## Reading 2 of the KO samples
raw_data <- readMSData2(fls[1:2])

## Perform the chromatographic peak detection using the matchedFilter method.
mfp <- MatchedFilterParam(snthresh = 20, binSize = 1)
res <- findChromPeaks(raw_data, param = mfp)

head(chromPeaks(res))
## The number of peaks identified per sample:
table(chromPeaks(res)[, "sample"])

## Performing the chromatographic peak grouping
fdp <- PeakDensityParam()
res <- groupChromPeaks(res, fdp)

## The definition of the features (peak groups): featureDefinitions(res)

## Using the featureValues method to extract a matrix with the intensities of the features per sample.
head(featureValues(res, value = "into"))

## The process history:
processHistory(res)
```
High resolution peak grouping for single spectra samples

Description

This method performs high resolution correspondence for single spectra samples.
The MzClustParam class allows to specify all settings for the peak grouping based on the mzClust algorithm. Instances should be created with the MzClustParam constructor.

Usage

```r
MzClustParam(sampleGroups = numeric(), ppm = 20, absMz = 0, minFraction = 0.5, minSamples = 1)
```

```r
## S4 method for signature 'MzClustParam'
show(object)
```

```r
## S4 method for signature 'MzClustParam'
sampleGroups(object)
```

```r
## S4 replacement method for signature 'MzClustParam'
sampleGroups(object) <- value
```

```r
## S4 method for signature 'MzClustParam'
ppm(object)
```

```r
## S4 replacement method for signature 'MzClustParam'
ppm(object) <- value
```

```r
## S4 method for signature 'MzClustParam'
absMz(object)
```

```r
## S4 replacement method for signature 'MzClustParam'
absMz(object) <- value
```

```r
## S4 method for signature 'MzClustParam'
minFraction(object)
```

```r
## S4 replacement method for signature 'MzClustParam'
minFraction(object) <- value
```
## S4 method for signature 'MzClustParam'
minSamples(object)

## S4 replacement method for signature 'MzClustParam'
minSamples(object) <- value

## S4 method for signature 'XCMSnExp,MzClustParam'
groupChromPeaks(object, param)

### Arguments

- **sampleGroups**: A vector of the same length than samples defining the sample group assignments (i.e. which samples belong to which sample group).
- **ppm**: numeric(1) representing the relative mz error for the clustering/grouping (in parts per million).
- **absMz**: numeric(1) representing the absolute mz error for the clustering.
- **minFraction**: numeric(1) defining the minimum fraction of samples in at least one sample group in which the peaks have to be present to be considered as a peak group (feature).
- **minSamples**: numeric(1) with the minimum number of samples in at least one sample group in which the peaks have to be detected to be considered a peak group (feature).
- **object**: For groupChromPeaks: an XCMSnExp object containing the results from a previous chromatographic peak detection analysis (see findChromPeaks). For all other methods: a MzClustParam object.
- **value**: The value for the slot.
- **param**: A MzClustParam object containing all settings for the peak grouping algorithm.

### Value

The MzClustParam function returns a MzClustParam class instance with all of the settings specified for high resolution single spectra peak alignment.

For groupChromPeaks: a XCMSnExp object with the results of the peak grouping step (i.e. the features). These can be accessed with the FeatureDefinitions method.

### Slots

- __classVersion__, sampleGroups, ppm, absMz, minFraction, minSamples

See corresponding parameter above. __classVersion__ stores the version from the class. Slots values should exclusively be accessed via the corresponding getter and setter methods listed above.

### Note

These methods and classes are part of the updated and modernized xcms user interface which will eventually replace the group methods. All of the settings to the algorithm can be passed with a MzClustParam object.

Calling groupChromPeaks on an XCMSnExp object will cause all eventually present previous correspondence results to be dropped.
References

Saira A. Kazmi, Samiran Ghosh, Dong-Guk Shin, Dennis W. Hill and David F. Grant
Alignment of high resolution mass spectra: development of a heuristic approach for metabolomics.

See Also

The do_groupPeaks_mzClust core API function and group.mzClust for the old user interface.
featureDefinitions and featureValues, XCMSnExp-method for methods to access peak grouping results (i.e. the features).
XCMSnExp for the object containing the results of the peak grouping.

Other peak grouping methods: groupChromPeaks-density, groupChromPeaks-nearest, groupChromPeaks

Examples

```r
## Loading a small subset of direct injection, single spectrum files
library(msdata)
fticrf <- list.files(system.file("fticr", package = "msdata"),
recursive = TRUE, full.names = TRUE)
fticr <- readMSData2(fticrf[1:2], msLevel. = 1)

## Perform the MSW peak detection on these:
p <- MSWParam(scales = c(1, 7), peakThr = 80000, ampTh = 0.005,
SNR.method = "data.mean", winSize.noise = 500)
fticr <- findChromPeaks(fticr, param = p)

head(chromPeaks(fticr))

## Now create the MzClustParam parameter object: we’re assuming here that
## both samples are from the same sample group.
p <- MzClustParam(sampleGroups = c(1, 1))
fticr <- groupChromPeaks(fticr, param = p)

## Get the definition of the features.
featureDefinitions(fticr)
```

---

**groupChromPeaks-nearest**

*Peak grouping based on proximity in the mz-rt space*

**Description**

This method is inspired by the grouping algorithm of mzMine [Katajamaa 2006] and performs correspondence based on proximity of peaks in the space spanned by retention time and mz values. The method creates first a master peak list consisting of all chromatographic peaks from the sample in which most peaks were identified, and starting from that, calculates distances to peaks from the sample with the next most number of peaks. If peaks are closer than the defined threshold they are grouped together.

The NearestPeaksParam class allows to specify all settings for the peak grouping based on the nearest algorithm. Instances should be created with the NearestPeaksParam constructor.
sampleGroups, sampleGroups<-: getter and setter for the sampleGroups slot of the object.
mzVsRtBalance, mzVsRtBalance<-: getter and setter for the mzVsRtBalance slot of the object.
absMz, absMz<-: getter and setter for the absMz slot of the object.
absRt, absRt<-: getter and setter for the absRt slot of the object.
kNN, kNN<-: getter and setter for the kNN slot of the object.

groupChromPeaks, XCMSnExp, NearestPeaksParam: performs peak grouping based on the proximity between chromatographic peaks from different samples in the mz-rt range.

Usage

NearestPeaksParam(sampleGroups = numeric(), mzVsRtBalance = 10, absMz = 0.2, absRt = 15, kNN = 10)

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

## S4 method for signature 'NearestPeaksParam'
sampleGroups(object)

## S4 replacement method for signature 'NearestPeaksParam'
sampleGroups(object) <- value

## S4 method for signature 'NearestPeaksParam'
mzVsRtBalance(object)

## S4 replacement method for signature 'NearestPeaksParam'
mzVsRtBalance(object) <- value

## S4 method for signature 'NearestPeaksParam'
absMz(object)

## S4 replacement method for signature 'NearestPeaksParam'
absMz(object) <- value

## S4 method for signature 'NearestPeaksParam'
absRt(object)

## S4 replacement method for signature 'NearestPeaksParam'
absRt(object) <- value

## S4 method for signature 'NearestPeaksParam'
kNN(object)

## S4 replacement method for signature 'NearestPeaksParam'
kNN(object) <- value

## S4 method for signature 'XCMSnExp, NearestPeaksParam'
groupChromPeaks(object, param)
groupChromPeaks-nearest

Arguments

 sampleGroups  A vector of the same length than samples defining the sample group assignments (i.e. which samples belong to which sample group).
 mZVsRtBalance  numeric(1) representing the factor by which mz values are multiplied before calculating the (euclidian) distance between two peaks.
 absMz  numeric(1) maximum tolerated distance for mz values.
 absRt  numeric(1) maximum tolerated distance for rt values.
 kNN  numeric(1) representing the number of nearest neighbors to check.
 object  For groupChromPeaks: an XCMSnExp object containing the results from a previous chromatographic peak detection analysis (see findChromPeaks).
 For all other methods: a NearestPeaksParam object.

Value

The NearestPeaksParam function returns a NearestPeaksParam class instance with all of the settings specified for peak alignment based on peak proximity.

For groupChromPeaks: a XCMSnExp object with the results of the peak grouping/correspondence step (i.e. the mz-rt features). These can be accessed with the featureDefinitions method.

Slots

__.classVersion__, sampleGroups, mZVsRtBalance, absMz, absRt, kNN  See corresponding parameter above.__.classVersion__ stores the version from the class. Slots values should exclusively be accessed via the corresponding getter and setter methods listed above.

Note

These methods and classes are part of the updated and modernized xcms user interface which will eventually replace the group methods. All of the settings to the algorithm can be passed with a NearestPeaksParam object.

Calling groupChromPeaks on an XCMSnExp object will cause all eventually present previous alignment results to be dropped.

References


See Also

The do_groupChromPeaks_nearest core API function and group.nearest for the old user interface. featureDefinitions and featureValues, XCMSnExp-method for methods to access peak grouping results (i.e. the features).

XCMSnExp for the object containing the results of the peak grouping.

Other peak grouping methods: groupChromPeaks-density, groupChromPeaks-mzClust, groupChromPeaks-density.
Examples

```r
## Create a NearestPeaksParam object
p <- NearestPeaksParam(kNN = 3)
p

##########################################################
## Chromatography peak detection and grouping.
##
## Below we perform first a chromatographic peak detection (using the
## matchedFilter method) on some of the test files from the faahKO package
## followed by a peaks grouping using the "nearest" method.
library(faahKO)
library(MSnbase)
fls <- dir(system.file("cdf/KO", package = "faahKO"), recursive = TRUE,
           full.names = TRUE)

## Reading 2 of the KO samples
raw_data <- readMSData2(fls[1:2])

## Perform the peak detection using the matchedFilter method.
mfp <- MatchedFilterParam(snthresh = 20, binSize = 1)
res <- findChromPeaks(raw_data, param = mfp)

head(chromPeaks(res))
## The number of peaks identified per sample:
table(chromPeaks(res)[, "sample"])

## Performing the peak grouping
p <- NearestPeaksParam()
res <- groupChromPeaks(res, param = p)

## The results from the peak grouping:
featureDefinitions(res)

## Using the featureValues method to extract a matrix with the intensities of
## the features per sample.
head(featureValues(res, value = "into"))

## The process history:
processHistory(res)
```

### groupnames-methods

*Generate unique names for peak groups*

**Description**

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

**Arguments**

- `object` : the xcmsSet or xcmsEIC object
- `mzdec` : number of decimal places to use for m/z
groupval-methods

rtdec number of decimal places to use for retention time
template a character vector with existing group names whose format should be emulated

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

Methods
\texttt{object = "xcmsSet"} (object, mzdec = 0, rtdec = 0, template = NULL)
\texttt{object = "xcmsEIC"} (object)

See Also
\texttt{xcmsSet-class, xcmsEIC-class}

---

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

Value
A matrix with with rows for every group and columns for every sample. Missing peaks have \texttt{NA} values.

Methods
\texttt{object = "xcmsSet"} groupval(object, method = c("medret", "maxint"), value = "index", intensity = "maxint")

See Also
\texttt{xcmsSet-class}
image-methods

Plot log intensity image of a xcmsRaw object

Description

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

Arguments

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

Methods

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

Author(s)

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

See Also

xcmsRaw-class

---

imputeLinInterpol

Impute values for empty elements in a vector using linear interpolation

Description

This function provides missing value imputation based on linear interpolation and resembles some of the functionality of the profBinLin and profBinLinBase functions deprecated from version 1.51 on.

Usage

imputeLinInterpol(x, baseValue, method = "lin", distance = 1L,
                  noInterpolAtEnds = FALSE)

Arguments

x         A numeric vector with eventual missing (NA) values.
baseValue  The base value to which empty elements should be set. This is only considered for method = "linbase" and corresponds to the profBinLinBase's baselevel argument.
method     One of "none", "lin" or "linbase".
distance   For method = "linbase": number of non-empty neighboring element of an empty element that should be considered for linear interpolation. See details section for more information.
noInterpolAtEnds
For method = "lin": Logical indicating whether linear interpolation should also be performed at the ends of the data vector (i.e. if missing values are present at the beginning or the end of the vector).

Details
Values for NAs in input vector x can be imputed using methods "lin" and "linbase":

impute = "lin" uses simple linear imputation to derive a value for an empty element in input vector x from its neighboring non-empty elements. This method is equivalent to the linear interpolation in the profBinLin method. Whether interpolation is performed if missing values are present at the beginning and end of x can be set with argument noInterpolAtEnds. By default interpolation is also performed at the ends interpolating from 0 at the beginning and towards 0 at the end. For noInterpolAtEnds = TRUE no interpolation is performed at both ends replacing the missing values at the beginning and/or the end of x with 0.

impute = "linbase" uses linear interpolation to impute values for empty elements within a user-definable proximity to non-empty elements and setting the element’s value to the baseValue otherwise. The default for the baseValue is half of the smallest value in x (NAs being removed). Whether linear interpolation based imputation is performed for a missing value depends on the distance argument. Interpolation is only performed if one of the next distance closest neighbors to the current empty element has a value other than NA. No interpolation takes place for distance = 0, while distance = 1 means that the value for an empty element is interpolated from directly adjacent non-empty elements while, if the next neighbors of the current empty element are also NA, it’s value is set to baseValue. This corresponds to the linear interpolation performed by the profBinLinBase method. For more details see examples below.

Value
A numeric vector with empty values imputed based on the selected method.

Author(s)
Johannes Rainer

Examples

```
###
## Impute missing values by linearly interpolating from neighboring
## non-empty elements
x <- c(3, NA, 1, 2, NA, NA, 4, NA, NA, 3, NA, NA, NA, NA, 2)
imputeLinInterpol(x, method = "lin")
## visualize the interpolation:
plot(x = 1:length(x), y = x)
points(x = 1:length(x), y = imputeLinInterpol(x, method = "lin"), type = "l", col = "grey")

## If the first or last elements are NA, interpolation is performed from 0
## to the first non-empty element.
x <- c(NA, 2, 1, 4, NA)
imputeLinInterpol(x, method = "lin")
## visualize the interpolation:
plot(x = 1:length(x), y = x)
points(x = 1:length(x), y = imputeLinInterpol(x, method = "lin"), type = "l", col = "grey")

## If noInterpolAtEnds is TRUE no interpolation is performed at both ends
```
levelplot-methods

Plot log intensity image of a xcmsRaw object

Description

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

Arguments

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

Methods

x = "xcmsRaw" levelplot(x, log=TRUE, col.regions=colorRampPalette(brewer.pal(9, "YlOrRd")))

x = "xcmsSet" levelplot(x, log=TRUE, col.regions=colorRampPalette(brewer.pal(9, "YlOrRd")))
Description

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

Arguments

object Specification of a data source (such as a file name or database query)

Details

The implementing methods decide how to gather the data.

Value

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

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

Methods

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

Author(s)

Daniel Hackney, <dan@haxney.org>

See Also

xcmsRaw-class, xcmsSource
**medianFilter**  
Apply a median filter to a matrix

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

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

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

**Value**
A matrix whose values have been median filtered

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

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

---

**MsFeatureData-class**  
Data container storing xcms preprocessing results

**Description**
The MsFeatureData class is designed to encapsulate all data related to the preprocessing of metabolomics data using the xcms package, i.e. it contains a matrix with the chromatographic peaks identified by the peak detection, a DataFrame with the definition on grouped chromatographic peaks across samples and a list with the adjusted retention times per sample. Along with the results, the object contains the processing history that allow to track each processing step along with the used settings. The object also directly extends the OnDiskMSnExp object hence allowing easy access to the full data on which the peak detection was performed.
Objects from this class should not be created directly, they are returned as result from the `findChromPeaks` method.

`XCMSnExp` objects can be coerced into `xcmsSet` objects using the `as` method.

`processHistoryTypes` returns the available types of process histories. These can be passed with argument `type` to the `processHistory` method to extract specific process step(s).

`profMat`: creates a profile matrix, which is a n x m matrix, n (rows) representing equally spaced m/z values (bins) and m (columns) the retention time of the corresponding scans. Each cell contains the maximum intensity measured for the specific scan and m/z values. See `profMat` for more details and description of the various binning methods.

`hasAdjustedRtime`: whether the object provides adjusted retention times.

`hasFeatures`: whether the object contains correspondence results (i.e. features).

`hasChromPeaks`: whether the object contains peak detection results.

`adjustedRtime`, `adjustedRtime<-`: extract/set adjusted retention times. `adjustedRtime<-` should not be called manually, it is called internally by the `adjustRtime` methods. For `XCMSnExp` objects, `adjustedRtime<-` does also apply the retention time adjustment to the chromatographic peaks in the object. The `bySample` parameter allows to specify whether the adjusted retention time should be grouped by sample (file).

`featureDefinitions`, `featureDefinitions<-`: extract or set the correspondence results, i.e. the mz-rt features (peak groups).

`chromPeaks`, `chromPeaks<-`: extract or set the matrix containing the information on identified chromatographic peaks. Parameter `bySample` allows to specify whether peaks should be returned ungrouped (default `bySample = FALSE`) or grouped by sample (`bySample = TRUE`). The `chromPeaks<-` method for `XCMSnExp` objects removes also all correspondence (peak grouping) and retention time correction (alignment) results. The optional arguments `rt`, `mz` and `ppm` allow to extract only chromatographic peaks overlapping (if `type = "any"`) or completely within (if `type = "within"`) the defined retention time and mz ranges. See description of the return value for details on the returned matrix. Users usually don’t have to use the `chromPeaks<-` method directly as detected chromatographic peaks are added to the object by the `findChromPeaks` method.

`rtime`: extracts the retention time for each scan. The `bySample` parameter allows to return the values grouped by sample/file and adjusted whether adjusted or raw retention times should be returned. By default the method returns adjusted retention times, if they are available (i.e. if retention times were adjusted using the `adjustRtime` method).

`mz`: extracts the mz values from each scan of all files within an `XCMSnExp` object. These values are extracted from the original data files and eventual processing steps are applied on the fly. Using the `bySample` parameter it is possible to switch from the default grouping of mz values by spectrum/scan to a grouping by sample/file.

`intensity`: extracts the intensity values from each scan of all files within an `XCMSnExp` object. These values are extracted from the original data files and eventual processing steps are applied on the fly. Using the `bySample` parameter it is possible to switch from the default grouping of intensity values by spectrum/scan to a grouping by sample/file.

`spectra`: extracts the `Spectrum` objects containing all data from object. The values are extracted from the original data files and eventual processing steps are applied on the fly. By setting `bySample = TRUE`, the spectra are returned grouped by sample/file. If the `XCMSnExp` object contains adjusted retention times, these are returned by default in the `Spectrum` objects (can be overwritten by setting `adjusted = FALSE`).

`processHistory`: returns a list with `ProcessHistory` objects (or objects inheriting from this base class) representing the individual processing steps that have been performed, eventually along
with their settings (Param parameter class). Optional arguments fileIndex and type allow to restrict to process steps of a certain type or performed on a certain file.

dropChromPeaks: drops any identified chromatographic peaks and returns the object without that information. Note that for XCMSnExp objects the method drops all results from a correspondence (peak grouping) or alignment (retention time adjustment) too. For XCMSnExp objects the method drops also any related process history steps.

dropFeatureDefinitions: drops the results from a correspondence (peak grouping) analysis, i.e. the definition of the mz-rt features and returns the object without that information. Note that for XCMSnExp objects the method will also drop retention time adjustment results, if these were performed after the last peak grouping (i.e. which base on the results from the peak grouping that are going to be removed). For XCMSnExp objects also all related process history steps are removed. Also eventually filled in peaks (by fillChromPeaks) will be removed too.

dropAdjustedRtime: drops any retention time adjustment information and returns the object without adjusted retention time. For XCMSnExp object this also reverts the retention times reported for the chromatographic peaks in the peak matrix to the original, raw, ones (after chromatographic peak detection). Note that for XCMSnExp objects the method drops also all peak grouping results if these were performed after the retention time adjustment. For XCMSnExp objects the method drops also any related process history steps.

dropFilledChromPeaks: drops any filled-in chromatographic peaks (filled in by the fillChromPeaks method) and all related process history steps.

Usage

processHistoryTypes()

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

## S4 method for signature 'MsFeatureData'
hasAdjustedRtime(object)

## S4 method for signature 'MsFeatureData'
hasFeatures(object)

## S4 method for signature 'MsFeatureData'
hasChromPeaks(object)

## S4 method for signature 'MsFeatureData'
adjustedRtime(object)

## S4 replacement method for signature 'MsFeatureData'
adjustedRtime(object) <- value

## S4 method for signature 'MsFeatureData'
dropAdjustedRtime(object)

## S4 method for signature 'MsFeatureData'
featureDefinitions(object)

## S4 replacement method for signature 'MsFeatureData'
featureDefinitions(object) <- value
MsFeatureData-class

## S4 method for signature 'MsFeatureData'
dropFeatureDefinitions(object)

## S4 method for signature 'MsFeatureData'
chromPeaks(object)

## S4 replacement method for signature 'MsFeatureData'
chromPeaks(object) <- value

## S4 method for signature 'MsFeatureData'
dropChromPeaks(object)

## S4 method for signature 'OnDiskMSnExp'
profMat(object, method = "bin", step = 0.1,
baselevel = NULL, basespace = NULL, mzrange. = NULL, fileIndex, ...)

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

## S4 method for signature 'XCMSnExp'
hasAdjustedRtime(object)

## S4 method for signature 'XCMSnExp'
hasFeatures(object)

## S4 method for signature 'XCMSnExp'
hasChromPeaks(object)

## S4 method for signature 'XCMSnExp'
adjustedRtime(object, bySample = FALSE)

## S4 replacement method for signature 'XCMSnExp'
adjustedRtime(object) <- value

## S4 method for signature 'XCMSnExp'
featureDefinitions(object)

## S4 replacement method for signature 'XCMSnExp'
featureDefinitions(object) <- value

## S4 method for signature 'XCMSnExp'
chromPeaks(object, bySample = FALSE, rt = numeric(),
mz = numeric(), ppm = 10, type = "any")

## S4 replacement method for signature 'XCMSnExp'
chromPeaks(object) <- value

## S4 method for signature 'XCMSnExp'
rtime(object, bySample = FALSE,
adjusted = hasAdjustedRtime(object))
## S4 method for signature 'XCMSnExp'
mz(object, bySample = FALSE, BPPARAM = bpparam())

## S4 method for signature 'XCMSnExp'
intensity(object, bySample = FALSE, BPPARAM = bpparam())

## S4 method for signature 'XCMSnExp'
spectra(object, bySample = FALSE, adjusted = hasAdjustedRtime(object), BPPARAM = bpparam())

## S4 method for signature 'XCMSnExp'
processHistory(object, fileIndex, type)

## S4 method for signature 'XCMSnExp'
donChromPeaks(object)

## S4 method for signature 'XCMSnExp'
donFeatureDefinitions(object, keepAdjRtime = FALSE, dropLastN = -1)

## S4 method for signature 'XCMSnExp'
dropAdjustedRtime(object)

## S4 method for signature 'XCMSnExp'
profMat(object, method = "bin", step = 0.1, baselevel = NULL, basespace = NULL, mzrange. = NULL, fileIndex, ...)

## S4 method for signature 'XCMSnExp,ANY'
findChromPeaks(object, param, BPPARAM = bpparam(), return.type = "XCMSnExp")

## S4 method for signature 'XCMSnExp'
donFilledChromPeaks(object)

### Arguments

**object**  
For adjustedRtime, featureDefinitions, chromPeaks, hasAdjustedRtime, hasFeatures and hasChromPeaks either a MsFeatureData or a XCMSnExp object, for all other methods a XCMSnExp object.

**value**  
For adjustedRtime <- a list (length equal to the number of samples) with numeric vectors representing the adjusted retention times per scan.  
For featureDefinitions <- a DataFrame with peak grouping information.  
See return value for the featureDefinitions method for the expected format.  
For chromPeaks <- a matrix with information on detected peaks. See return value for the chromPeaks method for the expected format.

**method**  
The profile matrix generation method. Allowed are "bin", "binlin", "binlinbase" and "intlin". See details section for more information.

**step**  
numeric(1) representing the m/z bin size.

**baselevel**  
numeric(1) representing the base value to which empty elements (i.e. m/z bins without a measured intensity) should be set. Only considered if method = "binlinbase". See baseValue parameter of imputeLinInterpol for more details.
basespace numeric(1) representing the m/z length after which the signal will drop to the base level. Linear interpolation will be used between consecutive data points falling within 2 * basespace to each other. Only considered if method = "binlinbase". If not specified, it defaults to 0.075. Internally this parameter is translated into the distance parameter of the imputeLinInterpol function by distance = floor(basespace / s) See distance parameter of imputeLinInterpol for more details.

mzrange. Optional numeric(2) manually specifying the m/z value range to be used for binning. If not provided, the whole m/z value range is used.

fileIndex For processHistory: optional numeric specifying the index of the files/samples for which the ProcessHistory objects should be retrieved.

... Additional parameters.

bySample logical(1) specifying whether results should be grouped by sample.

rt optional numeric(2) defining the retention time range for which chromatographic peaks should be returned.

mz optional numeric(2) defining the mz range for which chromatographic peaks should be returned.

ppm optional numeric(1) specifying the ppm by which the mz range should be extended. For a value of ppm = 10, all peaks within mz[1] - ppm / 1e6 and mz[2] + ppm / 1e6 are returned.

type For processHistory: restrict returned ProcessHistory objects to analysis steps of a certain type. Use the processHistoryTypes to list all supported values. For chromPeaks: character specifying which peaks to return if rt or mz are defined. For type = "any" all chromatographic peaks that overlap the range defined by the mz or by the rt. For type = "within" only peaks completely within the range(s) are returned.

adjusted logical(1) whether adjusted or raw (i.e. the original retention times reported in the files) should be returned.

BPPARAM Parameter class for parallel processing. See bpparam.

keepAdjRtime For dropFeatureDefinitions, XCMSnExp: logical(1) defining whether eventually present retention time adjustment should not be dropped. By default dropping feature definitions drops retention time adjustment results too.

dropLastN For dropFeatureDefinitions, XCMSnExp: numeric(1) defining the number of peak grouping related process history steps to remove. By default dropLastN = -1, dropping the chromatographic peaks removes all process history steps related to peak grouping. Setting e.g. dropLastN = 1 will only remove the most recent peak grouping related process history step.

param A CentWaveParam, MatchedFilterParam, MassifquantParam, MSWParam or CentWavePredIsoParam object with the settings for the chromatographic peak detection algorithm.

return.type Character specifying what type of object the method should return. Can be either "XCMSnExp" (default), "list" or "xcmsSet".

Value

For profMat: a list with a the profile matrix matrix (or matrices if fileIndex was not specified or if length(fileIndex) > 1). See profile-matrix for general help and information about the profile matrix.

For adjustedRtime: if bySample = FALSE a numeric vector with the adjusted retention for each spectrum of all files/samples within the object. If bySample = TRUE a list (length equal to the
number of samples) with adjusted retention times grouped by sample. Returns NULL if no adjusted retention times are present.

For featureDefinitions: a DataFrame with peak grouping information, each row corresponding to one mz-rt feature (grouped peaks within and across samples) and columns "mzmed" (median mz value), "mzmin" (minimal mz value), "mzmax" (maximum mz value), "rtmed" (median retention time), "rtmin" (minimal retention time), "rtmax" (maximal retention time) and "peakidx". Column "peakidx" contains a list with indices of chromatographic peaks (rows) in the matrix returned by the chromatPeaks method that belong to that feature group. The method returns NULL if no feature definitions are present.

For chromatPeaks: if bySample = FALSE a matrix with at least the following columns: "mz" (intensity-weighted mean of mz values of the peak across scans/retention times), "mzmin" (minimal mz value), "mzmax" (maximal mz value), "rt" (retention time for the peak apex), "rtmin" (minimal retention time), "rtmax" (maximal retention time), "into" (integrated, original, intensity of the peak), "maxo" (maximum intensity of the peak), "sample" (sample index in which the peak was identified) and "is_filled" defining whether the chromatographic peak was identified by the peak picking algorithm (0) or was added by the fillChromPeaks method (1). Depending on the employed peak detection algorithm and the verboseColumns parameter of it additional columns might be returned. For bySample = TRUE the chromatographic peaks are returned as a list of matrices, each containing the chromatographic peaks of a specific sample. For samples in which no peaks were detected a matrix with 0 rows is returned.

For rtime: if bySample = FALSE a numeric vector with the retention times of each scan, if bySample = TRUE a list of numeric vectors with the retention times per sample.

For mz: if bySample = FALSE a list with the mz values (numeric vectors) of each scan. If bySample = TRUE a list with the mz values per sample.

For intensity: if bySample = FALSE a list with the intensity values (numeric vectors) of each scan. If bySample = TRUE a list with the intensity values per sample.

For spectra: if bySample = FALSE a list with Spectrum objects. If bySample = TRUE the result is grouped by sample, i.e. as a list of lists, each element in the outer list being the list of spectra of the specific file.

For processHistory: a list of ProcessHistory objects providing the details of the individual data processing steps that have been performed.

Slots

- .processHistory list with XProcessHistory objects tracking all individual analysis steps that have been performed.
- msFeatureData MsFeatureData class extending environment and containing the results from a chromatographic peak detection (element "chromPeaks"), peak grouping (element "featureDefinitions") and retention time correction (element "adjustedRtime") steps.

Note

The "chromPeaks" element in the msFeatureData slot is equivalent to the @peaks slot of the xcmsSet object, the "featureDefinitions" contains information from the @groups and @groupidx slots from an xcmsSet object.

Author(s)

Johannes Rainer
See Also

- `xcmsSet` for the old implementation. `OnDiskMSnExp, MSnExp` and `pSet` for a complete list of inherited methods. `findChromPeaks` for available peak detection methods returning a `XCMSnExp` object as a result. `groupChromPeaks` for available peak grouping methods and `featureDefinitions` for the method to extract the feature definitions representing the peak grouping results. `adjustRtime` for retention time adjustment methods.

- `fillChromPeaks` for the method to fill-in eventually missing chromatographic peaks for a feature in some samples.

Examples

```r
## Loading the data from 2 files of the faahKO package.
library(faahKO)
od <- readMSData2(c(system.file("cdf/KO/ko15.CDF", package = "faahKO"),
                   system.file("cdf/KO/ko16.CDF", package = "faahKO")))
## Now we perform a chromatographic peak detection on this data set using the
## matched filter method. We are tuning the settings such that it performs
## faster.
mfp <- MatchedFilterParam(binSize = 4)
xod <- findChromPeaks(od, param = mfp)

## The results from the peak detection are now stored in the XCMSnExp
## object
xod

## The detected peaks can be accessed with the chromPeaks method.
head(chromPeaks(xod))

## The settings of the chromatographic peak detection can be accessed with
## the processHistory method
processHistory(xod)

## Also the parameter class for the peak detection can be accessed
processParam(processHistory(xod)[[1]])

## The XCMSnExp inherits all methods from the pSet and OnDiskMSnExp classes
## defined in Bioconductor's MSnbase package. To access the (raw) retention
## time for each spectrum we can use the rtime method. Setting bySample = TRUE
## would cause the retention times to be grouped by sample
head(rtime(xod))

## Similarly it is possible to extract the mz values or the intensity values
## using the mz and intensity method, respectively, also with the option to
## return the results grouped by sample instead of the default, which is
## grouped by spectrum. Finally, to extract all of the data we can use the
## spectra method which returns Spectrum objects containing all raw data.
## Note that all these methods read the information from the original input
## files and subsequently apply eventual data processing steps to them.
head(mz(xod, bySample = TRUE))

## Reading all data
spctr <- spectra(xod)
## To get all spectra of the first file we can split them by file
head(split(spctr, fromFile(xod))[[1]])
```
### Filtering

XCMSnExp objects can be filtered by file, retention time, mz values or MS level. For some of these filter preprocessing results (mostly retention time correction and peak grouping results) will be dropped. Below we filter the XCMSnExp object by file to extract the results for only the second file.

```r
xod_2 <- filterFile(xod, file = 2)
xod_2
```

Now the objects contains only the identified peaks for the second file

```r
head(chromPeaks(xod_2))
```

### Coercing to an xcmsSet object

We can also coerce the XCMSnExp object into an xcmsSet object:

```r
xs <- as(xod, "xcmsSet")
head(peaks(xs))
```

---

**msn2xcmsRaw**  
*Copy MSn data in an xcmsRaw to the MS slots*

**Description**

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

**Usage**

```r
msn2xcmsRaw(xmsn)
```

**Arguments**

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

**Details**

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

**Value**

An xcmsRaw object

**Author(s)**

Steffen Neumann <sneumann@ipb-halle.de>
See Also

xcmsRaw.

Examples

msnfile <- system.file("microtofq/MSMSpos20_6.mzML", package = "msdata")
xrmsn <- xcmsRaw(msnfile, includeMSN=TRUE)
xr <- msn2xcmsRaw(xrmsn)
p <- findPeaks(xr, method="centWave")
Description

Create a report showing all aligned peaks.

Arguments

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

Details

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

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

Value

A data frame with the following columns:

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

Methods

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

See Also

xcmsSet-class,
Examples

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

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

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

### Arguments
- **x**
  - the xcmsEIC object
- **y**
  - optional xcmsSet object with peak integration data
- **groupidx**
  - either character vector with names or integer vector with indicies of peak groups for which to plot EICs
- **sampleidx**
  - either character vector with names or integer vector with indicies of samples for which to plot EICs
- **rrange**
  - 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,

plotAdjustedRtime Visualization of alignment results

Description

Plot the difference between the adjusted and the raw retention time (y-axis) for each file along the (adjusted or raw) retention time (x-axis). If alignment was performed using the adjustRtime-peakGroups method, also the features (peak groups) used for the alignment are shown.

Usage

`plotAdjustedRtime(object, col = "#00000080", lty = 1, type = "l", adjustedRtime = TRUE, xlab = ifelse(adjustedRtime, yes = expression(rt[adj]), no = expression(rt[raw])), ylab = expression(rt[adj] - rt[raw]), peakGroupsCol = "#0000060", peakGroupsPch = 16, peakGroupsLty = 3, ...)

Arguments

object A `XCMSnExp` object with the alignment results.

col colors to be used for the lines corresponding to the individual samples.

lty line type to be used for the lines of the individual samples.

type plot type to be used. See help on the par function for supported values.

adjustedRtime logical(1) whether adjusted or raw retention times should be shown on the x-axis.

xlab the label for the x-axis.

ylab the label for the y-axis.

peakGroupsCol color to be used for the peak groups (only used if alignment was performed using the adjustRtime-peakGroups method).

peakGroupsPch point character (pch) to be used for the peak groups (only used if alignment was performed using the adjustRtime-peakGroups method).

peakGroupsLty line type (lty) to be used to connect points for each peak groups (only used if alignment was performed using the adjustRtime-peakGroups method).

... Additional arguments to be passed down to the plot function.

Author(s)

Johannes Rainer
See Also

adjustRtime for all retention time correction/alignment methods.

Examples

## Below we perform first a peak detection (using the matchedFilter
## method) on some of the test files from the faahKO package followed by
## a peak grouping and retention time adjustment using the “peak groups”
## method
library(faahKO)
library(xcms)
fls <- dir(system.file("cdf/KO", package = "faahKO"), recursive = TRUE,
        full.names = TRUE)

## Reading 2 of the KO samples
raw_data <- readMSData2(fls[1:2])

## Perform the peak detection using the matchedFilter method.
mfp <- MatchedFilterParam(snthresh = 20, binSize = 1)
res <- findChromPeaks(raw_data, param = mfp)

## Performing the peak grouping using the "peak density" method.
p <- PeakDensityParam(sampleGroups = c(1, 1))
res <- groupChromPeaks(res, param = p)

## Perform the retention time adjustment using peak groups found in both
## files.
fgp <- PeakGroupsParam(minFraction = 1)
res <- adjustRtime(res, param = fgp)

## Visualize the impact of the alignment. We show both versions of the plot,
## with the raw retention times on the x-axis (top) and with the adjusted
## retention times (bottom).
par(mfrow = c(2, 1))
plotAdjustedRtime(res, adjusted = FALSE)
grid()
plotAdjustedRtime(res)
grid()

plotChrom-methods  

Plot extracted ion chromatograms from the profile matrix

Description

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

Arguments

object  

the xcmsRaw object

base  

logical, plot a base-peak chromatogram

ident  

logical, use mouse to identify and label peaks

fitgauss  

logical, fit a gaussian to the largest peak
Value

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

Methods

```
object = "xcmsRaw" plotEIC(object, mzrange = numeric(), rrange = numeric(), scanrange = numeric(), mzdec=2, type="l", add=FALSE, ...) 
```

See Also

```
xcmsRaw-class
```

---

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

### Description

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

### Arguments

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

### Value

A two-column matrix with the plotted points.

### Methods

```
object = "xcmsRaw" plotEIC(object, mzrange = numeric(), rrange = numeric(), scanrange = numeric(), mzdec=2, type="l", add=FALSE, ...) 
```

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

**Details**

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

**Methods**

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

**See Also**

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

---

**plotQC**

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

**Description**

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

**Usage**

```r
plotQC(object, sampNames, sampColors, sampOrder, what)
```
Arguments

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

Details

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

Value

List with four matrices, each of dimension features * samples: "mz": median m/z deviation for each sample "mzdev": median m/z deviation for each sample "rt": median RT deviation for each sample "rtdev": median RT deviation for each sample

Author(s)

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

Examples

```r
library(faahKO)
xsg <- group(faahko)

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

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.
**plotrt-methods**

**Arguments**

- **object**: the `xcmsRaw` 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`

---

**plotrt-methods**  

**Plot retention time deviation profiles**

**Description**

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

**Arguments**

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

**Methods**

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

**See Also**

- `xcmsSet-class, retcor`
### plotScan-methods

#### Plot a single mass scan

**Description**

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

**Arguments**

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

**Methods**

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

**See Also**

- `xcmsRaw-class`

### plotSpec-methods

#### Plot mass spectra from the profile matrix

**Description**

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

**Arguments**

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

**Value**

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

**Methods**

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

**See Also**

- `xcmsRaw-class`
plotSurf-methods

Description

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

Arguments

object the \texttt{xcmsRaw} object
log logical, log transform intensity
aspect numeric vector with aspect ratio of the m/z, retention time and intensity components of the plot
... arguments passed to \texttt{profRange}

Details

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

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

Methods

\texttt{object = "xcmsRaw"} plotSurf(object, log = \texttt{FALSE}, aspect = \texttt{c(1, 1, .5), ...})

See Also

\texttt{xcmsRaw-class}

plotTIC-methods

Description

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

Arguments

object the \texttt{xcmsRaw} object
ident logical, use mouse to identify and label chromatographic peaks
msident logical, use mouse to identify and label spectral peaks
Value

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

Methods

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

See Also

`xcmsRaw-class`

---

ProcessHistory-class  Tracking data processing

Description

Objects of the type `ProcessHistory` allow to keep track of any data processing step in an metabolomics experiment. They are created by the data processing methods, such as `findChromPeaks` and added to the corresponding results objects. Thus, usually, users don’t need to create them.

The `XProcessHistory` extends the `ProcessHistory` by adding a slot `param` that allows to store the actual parameter class of the processing step.

Get or set the parameter class from an `XProcessHistory` object.

The `processType` method returns a character specifying the processing step `type`.

The `processDate` extracts the start date of the processing step.

The `processInfo` extracts optional additional information on the processing step.

The `fileIndex` extracts the indices of the files on which the processing step was applied.

Usage

```r
## S4 method for signature 'ProcessHistory'
show(object)

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

## S4 method for signature 'XProcessHistory'
processParam(object)

## S4 method for signature 'ProcessHistory'
processType(object)

## S4 method for signature 'ProcessHistory'
processDate(object)

## S4 method for signature 'ProcessHistory'
processInfo(object)

## S4 method for signature 'ProcessHistory'
fileIndex(object)
```
profMat-xcmsSet

Arguments

object A ProcessHistory or XProcessHistory object.

Value

For processParam: a parameter object extending the Param class.

The processType method returns a character string with the processing step type.

The processDate method returns a character string with the time stamp of the processing step start.

The processInfo method returns a character string with optional additional informations.

The fileIndex method returns a integer vector with the index of the files/samples on which the processing step was applied.

Slots

type character(1): string defining the type of the processing step. This string has to match predefined values. Use processHistoryTypes to list them.

date character(1): date time stamp when the processing step was started.

info character(1): optional additional information.

fileIndex integer of length 1 or > 1 to specify on which samples of the object the processing was performed.

error (ANY): used to store eventual calculation errors.

param (Param): an object of type Param (e.g. CentWaveParam) specifying the settings of the processing step.

Author(s)

Johannes Rainer

profMat-xcmsSet  The profile matrix

Description

The profile matrix is an n x m matrix, n (rows) representing equally spaced m/z values (bins) and m (columns) the retention time of the corresponding scans. Each cell contains the maximum intensity measured for the specific scan and m/z values falling within the m/z bin.

The profMat method creates a new profile matrix or returns the profile matrix within the object’s @env slot, if available. Settings for the profile matrix generation, such as step (the bin size), method or additional settings are extracted from the respective slots of the xcmsRaw object. Alternatively it is possible to specify all of the settings as additional parameters.

Usage

## S4 method for signature 'xcmsRaw'
profMat(object, method, step, baselevel, basespace, mzrange.)
Arguments

object The \texttt{xcmsRaw} object.

method The profile matrix generation method. Allowed are "bin", "binlin", "binlinbase" and "intlin". See details section for more information.

step numeric(1) representing the m/z bin size.

baselevel numeric(1) representing the base value to which empty elements (i.e. m/z bins without a measured intensity) should be set. Only considered if method = "binlinbase". See baseValue parameter of \texttt{imputeLinInterpol} for more details.

basespace numeric(1) representing the m/z length after which the signal will drop to the base level. Linear interpolation will be used between consecutive data points falling within 2 * basespace to each other. Only considered if method = "binlinbase". If not specified, it defaults to 0.075. Internally this parameter is translated into the distance parameter of the \texttt{imputeLinInterpol} function by distance = floor(basespace / step). See distance parameter of \texttt{imputeLinInterpol} for more details.

mzrange Optional numeric(2) manually specifying the m/z value range to be used for binning. If not provided, the whole m/z value range is used.

Details

Profile matrix generation methods:

- **bin** The default profile matrix generation method that does a simple binning, i.e. aggregating of intensity values falling within an m/z bin.

- **binlin** Binning followed by linear interpolation to impute missing values. The value for m/z bins without a measured intensity are inferred by a linear interpolation between neighboring bins with a measured intensity.

- **binlinbase** Binning followed by a linear interpolation to impute values for empty elements (m/z bins) within a user-definable proximity to non-empty elements while setting the element’s value to the baselevel otherwise. See impute = "linbase" parameter of \texttt{imputeLinInterpol} for more details.

- **intlin** Set the elements’ values to the integral of the linearly interpolated data from plus to minus half the step size.

Value

\texttt{profMat} returns the profile matrix (rows representing scans, columns equally spaced m/z values).

Note

From \texttt{xcms} version 1.51.1 on only the \texttt{profMat} method should be used to extract the profile matrix instead of the previously default way to access it directly via \texttt{object@env$profile}.

Author(s)

Johannes Rainer

See Also

\texttt{xcmsRaw}, \texttt{binYonX} and \texttt{imputeLinInterpol} for the employed binning and missing value imputation methods, respectively. \texttt{profMat,XCMSnExp-method} for the method on \texttt{XCMSnExp} objects.
Examples

```r
file <- system.file('cdf/KO/ko15.CDF', package = "faahKO")
## Load the data without generating the profile matrix (profstep = 0)
xraw <- xcmsRaw(file, profstep = 0)
## Extract the profile matrix
profmat <- profMat(xraw, step = 0.3)
dim(profmat)
## If not otherwise specified, the settings from the xraw object are used:
profinfo(xraw)
## To extract a profile matrix with linear interpolation use
profmat <- profMat(xraw, step = 0.3, method = "binlin")
## Alternatively, the profMethod of the xraw objects could be changed
profMethod(xraw) <- "binlin"
profmat_2 <- profMat(xraw, step = 0.3)
all.equal(profmat, profmat_2)
```

profMedFilt-methods  Median filtering of the profile matrix

Description

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

Arguments

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

Methods

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

See Also

- `xcmsRaw-class`, `medianFilter`

profMethod-methods  Get and set method for generating profile data

Description

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

Methods

`object = "xcmsRaw"` profMethod(object)

See Also

- `xcmsRaw-class`, `profMethod`, `profBin`, `plotSpec`, `plotChrom`, `findPeaks`
Specify a subset of profile mode data

Description

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

Arguments

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

Details

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

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

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

Value

A list with the following items:

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

Methods

```
object = "xcmsRaw" profRange(object, mzrange = numeric(), rtrange = numeric(), scanrange = numeric())
```
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(00, 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 (i.e. values binned along the M/Z dimension).

Arguments

object xcmsRaw object
mzrange m/z range for EIC
rtrange retention time range for EIC
scanrange scan range for EIC
Value

A list of:

- scan: scan number
- intensity: added intensity values

Methods

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

Author(s)

Ralf Tautenhahn

See Also

- xcmsRaw-class

rawMat-methods  Get a raw data matrix

Description

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

Arguments

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

Value

A numeric matrix with three columns: time, m/z and intensity.

Methods

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

Author(s)

Michael Lawrence

See Also

- plotRaw for plotting the raw intensities
retcor-methods

Correct retention time from different samples

Description

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

Arguments

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

Details

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

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

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

Value

An xcmsSet object with corrected retention times.

Methods

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

See Also

retcor.loess retcor.obiwarp xcmsSet-class.

retcor.obiwarp

Align retention times across samples with Obiwp

Description

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

For the original publication see

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

object: the xcmsSet object
plottype: if deviation plot retention time deviation
profStep: step size (in m/z) to use for profile generation from the raw data files
center: the index of the sample all others will be aligned to. If center==NULL, the sample with the most peaks is chosen as default.
col: vector of colors for plotting each sample
ty: vector of line and point types for plotting each sample
response: Responsiveness of warping. 0 will give a linear warp based on start and end points. 100 will use all bijective anchors
distFunc: DistFunc function: cor (Pearson’s R) or cor_opt (default, calculate only 10% diagonal band of distance matrix, better runtime), cov (covariance), prd (product), euc (Euclidean distance)
gapInit: Penalty for Gap opening, see below
gapExtend: Penalty for Gap enlargement, see below
factorDiag: Local weighting applied to diagonal moves in alignment.
factorGap: Local weighting applied to gap moves in alignment.
localAlignment: Local rather than global alignment
initPenalty: Penalty for initiating alignment (for local alignment only) Default: 0
Default gap penalties: (gapInit, gapExtend) [by distFunc type]: ’cor’ = ’0.3,2.4’ ’cov’ = ’0.11.7’ ’prd’ = ’0.7.8’ ’euc’ = ’0.9,1.8’

Value

An xcmsSet object

Methods

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

See Also

xcmsSet-class.

Description

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

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

**Value**

An `xcmsSet` object

**Methods**

```r
object = "xcmsSet"  retcor(object, missing = 1, extra = 1, smooth = c("loess", "linear"),,

See Also

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

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

The altered matrix.

**Author(s)**

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

getEIC

Get sample names

Description

Return sample names for an object

Value

A character vector with sample names.

Methods

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

See Also

xcmsSet-class, xcmsEIC-class

showError, xcmsSet-method

Extract processing errors

Description

If peak detection is performed with findPeaks setting argument stopOnError = FALSE eventual errors during the process do not cause to stop the processing but are recorded inside of the resulting xcmsSet object. These errors can be accessed with the showError method.

Usage

## S4 method for signature 'xcmsSet'
showError(object, message. = TRUE, ...)

Arguments

object  An xcmsSet object.
message. Logical indicating whether only the error message, or the error itself should be returned.
...  Additional arguments.

Value

A list of error messages (if message. = TRUE) or errors or an empty list if no errors are present.

Author(s)

Johannes Rainer
## Description

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

## Arguments

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

## Details

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

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

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

## Value

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

## Methods

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

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

## Author(s)

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

Description

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

Usage

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

Arguments

- **peakTable1**: a Matrix containing at least m/z-values, row must be called "mz"
- **peakTable2**: the matrix for the other m/z-values
- **mzabs**: maximum absolute deviation for two matching peaks
- **mzppm**: relative deviations in ppm for two matching peaks
- **symmetric**: use symmetric pairwise m/z-matches only, or each match
- **mzExp**: the exponent used for m/z
- **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 m/z and intensity from matching peaks in the two peaktables. The factors are calculated as \( w_{\text{Fact}} = m/z^{\text{mzExp}} \times \text{int}^{\text{intExp}} \). If no distance is calculated (for example because no matching peaks were found) the return-value is NA.

Methods

peakTable1 = "matrix", peakTable2 = "matrix" specDist.cosine(peakTable1, peakTable2, mzabs = 0.001)

Author(s)

Joachim Kutzera, <jkutzer@ipb-halle.de>
specDist.meanMZmatch is a Distance function based on matching peaks.

**Description**

This method calculates the distance of two sets of peaks.

**Usage**

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

**Arguments**

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

**Details**

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

**Methods**

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

**Author(s)**

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

---

specDist.peakCount-methods is 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**

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

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

**Methods**

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

**Author(s)**

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

---

**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**

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

**Arguments**

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

**Details**

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

**Value**

A numeric noise level

**Author(s)**

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

**See Also**

`getSpec`, `specPeaks`
**specPeaks**

Identify peaks in a sparse continuum mode spectrum

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

A matrix with columns:

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

**Author(s)**

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

**See Also**

`getSpec`, `specNoise`
**split.xcmsRaw**  
*Divide an xcmsRaw object*

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

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

**Value**
A list of xcmsRaw objects.

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

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

**See Also**
- `xcmsRaw-class`

**split.xcmsSet**  
*Divide an xcmsSet object*

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

**Arguments**
- **xs**: xcmsSet object
- **f**: factor such that `factor(f)` defines the grouping
- **drop**: logical indicating if levels that do not occur should be dropped (if `f` is a `factor` or a list).
- **...**: further potential arguments passed to methods.
Description
This selfStart model evaluates the Gaussian model and its gradient. It has an initial attribute that will evaluate the initial estimates of the parameters mu, sigma, and h.

Usage
SSgauss(x, mu, sigma, h)

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

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

Value
A numeric vector of the same length as x. It is the value of the expression h*exp(-(x-mu)^2/(2*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>

See Also
nls, selfStart
stitch-methods

Correct gaps in data

Description

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

Arguments

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

Details

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

Value

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

Methods

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

Author(s)

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

Examples

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

Update an xcmsSet object

Description
This method updates an old xcmsSet object to the latest definition.

Usage
## S4 method for signature 'xcmsSet'
updateObject(object, ..., verbose = FALSE)

Arguments

object The xcmsSet object to update.

... Optional additional arguments. Currently ignored.

verbose Currently ignored.

Value
An updated xcmsSet containing all data from the input object.

Author(s)
Johannes Rainer

useOriginalCode Enable usage of old xcms code

Description
This function allows to enable the usage of old, partially deprecated code from xcms by setting a corresponding global option. See details for functions affected.

Usage
useOriginalCode(x)
Arguments

logical(1) to specify whether or not original old code should be used in corresponding functions. If not provided the function simply returns the value of the global option.

Details

The functions/methods that will be affected by this are:

• *do_findChromPeaks_matchedFilter*

Value

logical(1) indicating whether old code is being used.

Note

Usage of old code is strongly discouraged. This function is thought to be used mainly in the transition phase from xcms to xcms version 3.

Author(s)

Johannes Rainer

---

**verify.mzQuantM**

*Verify an mzQuantML file*

**Description**

Export in XML data formats: verify the written data

**Usage**

```r
define verify.mzQuantML(filename, xsdfilename)
```

**Arguments**

- `filename`: filename (may include full path) for the output file. Pipes or URLs are not allowed.
- `xsdfilename`: Filename of the XSD to verify against (may include full path)

**Details**

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

**Value**

None.

**See Also**

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

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

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

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

**Value**
None.

**Methods**

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

**See Also**
xcmsRaw-class, xcmsRaw.

---

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

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

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

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

None.

Methods

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

See Also

xcmsRaw-class, xcmsRaw

write.mzQuantML-methods

Save an xcmsSet object to an PSI mzQuantML file

Description

Export in XML data formats: Write the processed data in an xcmsSet to mzQuantML.

Arguments

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

Details

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

Value

None.

Methods

object = "xcmsSet" write.mzQuantML(object, filename)

See Also

xcmsSet-class, xcmsSet, verify.mzQuantML
writeMzTab  

Save a grouped xcmsSet object in mzTab-1.1 format file

Description

Write the grouped xcmsSet to an mzTab file.

Arguments

object  
the xcmsSet object

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

Details

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

Value

None.

Usage

object = "xcmsSet" writeMzTab(object, filename)

See Also

xcmsSet-class, xcmsSet.

Examples

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

Deprecated functions in package ‘xcms’

Description

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

Details

The following functions/methods are deprecated.

• `xcmsPapply`: this function is no longer available and the use of `bplapply` is suggested.
• `profBin`, `profBinM`, `profBinLin`, `profBinLinM`, `profBinLinBase`, `profBinLinBaseM` have been deprecated and `binYonX` in combination with `imputeLinInterpol` should be used in-

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
xcmsFileSource-class

Note
No notes yet.

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

See Also
getEIC

xcmsFileSource-class  Base class for loading raw data from a file

Description
Data sources which read data from a file should inherit from this class. The xcms package provides classes to read from netCDF, mzData, mzXML, and mzML files using xcmsFileSource. This class should be considered virtual and will not work if passed to loadRaw-methods. The reason it is not explicitly virtual is that there does not appear to be a way for a class to be both virtual and have a data part (which lets functions treat objects as if they were character strings). This class validates that a file exists at the path given.

Objects from the Class
xcmsFileSource objects should not be instantiated directly. Instead, create subclasses and instantiate those.

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

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

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

Author(s)
Daniel Hackney <dan@haxney.org>

See Also
xcmsSource
xcmsFragments

Constructor for xcmsFragments objects which holds Tandem MS peaks

Description

EXPERIMENTAL FEATURE

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

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

Usage

xcmsFragments(xs, ...)

Arguments

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

...
进一步 arguments to the collect method

Details

After running collect(xFragments,xSet) The peaktab 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
**xcmsFragments-class**  

*Class xcmsFragments, a class for handling Tandem MS and MS$^n$ data*

---

**Description**

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

**Objects from the Class**

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

**Slots**

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

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

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

**Methods**

- **collect** signature(object = "xcmsFragments"): gets a xcmsSet-object, collects ms1-peaks from it and the msn-peaks from the corresponding xcmsRaw-files.
- **plotTree** signature(object = "xcmsFragments"): prints a (text based) pseudo-tree of the peak-table to display the dependencies of the peaks among each other.
- **show** signature(object = "xcmsFragments"): print a human-readable description of this object to the console.

**Note**

No notes yet.

**Author(s)**

S. Neumann, J. Kutzera
References

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

See Also

xcmsRaw

xcmsPapply

Description

This function is deprecated, use bplapply instead.

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

Usage

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

Arguments

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

Details

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

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

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

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

- OpenMPI: Unix, http://www.open-mpi.org/, don’t forget to export MPI_ROOT before installing Rmpi e.g. export MPI_ROOT=/usr/lib/openmpi
\textbf{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`.

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

\textbf{Author(s)} \\
Duane Currie <duane.currie@acadiau.ca>, modified by Ralf Tautenhahn <rtautenh@ipb-halle.de>.

\textbf{References} \\
http://ace.acadiau.ca/math/ACMMaC/software/papply/

\textbf{Examples} \\
\begin{verbatim}
## 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)
\end{verbatim}

---

\textbf{xcmsPeaks-class} \\
A matrix of peaks

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

\textbf{Objects from the Class} \\
Objects can be created by calls of the form `new("xcmsPeaks", ...)`. 

\textbf{Slots} \\
\texttt{.Data}: The matrix holding the peak information
Extends


Methods

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

Author(s)

Michael Lawrence

See Also

findPeaks for detecting peaks in an xcmsRaw.

---

### xcmsRaw

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

#### Description

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

#### Usage

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

deepCopy(object)

#### Arguments

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

See profile-matrix for details on profile matrix generation methods and settings.

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

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

Value

A xcmsRaw object.

Author(s)

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

References

NetCDF file format: http://my.unidata.ucar.edu/content/software/netcdf/

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

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


See Also

xcmsRaw-class, profStep, profMethod xcmsFragments

Examples

## Not run:
library(xcms)
library(faahKO)
cdfpath <- system.file("cdf", package = "faahKO")
cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)
xr<-xcmsRaw(cdffiles[1])
xr
##This gives some information about the file
names(attributes(xr))
## Lets have a look at the structure of the object
str(xr)
##same but with a preview of each slot in the object
##50... 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]]
### 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`: Numeric representing the acquisition number of the individual scans/spectra. Length of `acquisitionNum` is equal to the number of spectra/scans in the object and hence equal to the `scantime` slot. Note however that this information is only available in mzML files.
- `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. The profile matrix should be extracted with the `profMat` method.
- `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

class: polarity

profMethod: character value with name of method used for generating the profile matrix.

profParam: list to store additional profile matrix generation settings. Use the profinfo method to extract all profile matrix creation relevant information.

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

msLevel: Numeric representing the MS level that is present in MS1 slot. This slot should be accessed through its getter method mslevel.

scanRange: Numeric of length 2 specifying the scan range (or NULL for the full range). This slot should be accessed through its getter method scanrange. Note that the scanRange will always be 1 to the number of scans within the xcmsRaw object, which does not necessarily have to match to the scan index in the original mzML file (e.g. if the original data was sub-setted). The acquisitionNum information can be used to track the original position of each scan in the mzML file.

Methods

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

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

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

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

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

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

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

msLevel Getter method for the msLevel slot.

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

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

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

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

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

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

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

profMedFilt signature(object = "xcmsRaw") median filter profile data in time and m/z dimensions
`profMethod<-` signature(object = "xcmsRaw"): change the method of generating the profile matrix
`profMethod` signature(object = "xcmsRaw"): get the method of generating the profile matrix
`profMz` signature(object = "xcmsRaw"): get vector of m/z values for each row of the profile matrix
`profRange` signature(object = "xcmsRaw"): interpret flexible ways of specifying subsets of the profile matrix
`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
`scanrange` Getter method for the `scanrange` slot. See slot description above for more information.
`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>, Johannes Rainer <johannes.rainer@eurac.edu>

**References**

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

**See Also**

`xcmsRaw, subset-xcmsRaw` for subsetting by spectra.

---

### `xcmsSet`

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

**Description**

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

**Usage**

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

files
  path names of the NetCDF/mzXML files to read

snames
  sample names. By default the file name without extension is used.

class
  sample classes.

phenoData
  data.frame or AnnotatedDataFrame defining the sample names and classes and other sample related properties. If not provided, the argument class or the subdirectories in which the samples are stored will be used to specify sample grouping.

profmethod
  Method to use for profile generation. Supported values are "bin", "binlin", "binlinbase" and "intlin" (for methods profBin, profBinLin, profBinLinBase and profIntLin, respectively). See help on profBin for a complete list of available methods and their supported parameters.

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
  DEPRECATED, use BPPARAM argument instead.

progressCallback
  function to be called, when progressInfo changes (useful for GUIs)

scanrange
  scan range to read

BPPARAM
  a BiocParallel parameter object to control how and if parallel processing should be performed. Such objects can be created by the SerialParam, MulticoreParam or SnowParam functions.

stopOnError
  Logical specifying whether the feature detection call should stop on the first encountered error (the default), or whether feature detection is performed in all files regardless eventual failures for individual files in which case all errors are reported as warnings.

... further arguments to the findPeaks method of the xcmsRaw class

Details

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

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

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

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

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

A `xcmsSet` object.

Note

The arguments `profmethod` and `profparam` have no influence on the feature/peak detection. The step size parameter `step` for the profile generation in the `findPeaks.matchedFilter` peak detection algorithm can be passed using the `...`.

Author(s)

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

See Also

`xcmsSet-class`, `findPeaks`, `profStep`, `profMethod`, `profBin`, `xcmsPapply`

---

### `xcmsSet-class`

**Class** `xcmsSet`, a class for preprocessing peak data

**Description**

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

**Details**

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

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

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

**Objects from the Class**

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

- **peaks** matrix containing peak data.
- **filled** A vector with peak indices of peaks which have been added by a `fillPeaks` method.
- **groups** Matrix containing statistics about peak groups.
- **groupidx** List containing indices of peaks in each group.
- **phenoData** A data.frame containing the experimental design factors.
- **rt** List containing two lists, raw and corrected, each containing retention times for every scan of every sample.
- **filepaths** Character vector with absolute path name of each NetCDF file.
- **profinfo** list containing the values method - profile generation method, and step - profile m/z step size and eventual additional parameters to the profile function.
- **dataCorrection** logical vector filled if the waters Lock mass correction parameter is used.
- **polarity** A string ("positive" or "negative" or NULL) describing whether only positive or negative scans have been used reading the raw data.
- **progressInfo** Progress informations for some xcms functions (for GUI).
- **progressCallback** Function to be called, when progressInfo changes (for GUI).
- **mslevel** Numeric representing the MS level on which the peak picking was performed (by default on MS1). This slot should be accessed through its getter method `mslevel`.
- **scanrange** Numeric of length 2 specifying the scan range (or NULL for the full range). This slot should be accessed through its getter method `scanrange`. The scan range provided in this slot represents the scans to which the whole raw data is subsetted.
- **.processHistory** Internal slot to be used to keep track of performed processing steps. This slot should not be directly accessed by the user.

Methods

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

**group** signature(object = "xcmsSet"): find groups of peaks across samples that share similar m/z and retention times

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

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

**peaks** signature(object = "xcmsSet"): get peaks slot

**plotrt** signature(object = "xcmsSet"): plot retention time deviation profiles

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

**profinfo** signature(object = "xcmsSet"): get profinfo slot

**profMethod** signature(object = "xcmsSet"): extract the method used to generate the profile matrix.

**profStep** signature(object = "xcmsSet"): extract the profile step used for the generation of the profile matrix.

**retcor** signature(object = "xcmsSet"): use initial grouping of peaks to do nonlinear loess retention time correction

**sampclass**<- signature(object = "xcmsSet"): Replaces the column “class” in the phenoData slot. See details for more information.

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

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

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

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

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

**scanrange** Getter method for the scanrange slot. See scanrange slot description above for more details.

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

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

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

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

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

**Note**

No notes yet.

**Author(s)**

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

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

See Also

xcmsSet

xcmsSource-class

Virtual class for raw data sources

Description

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

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

Objects from the Class

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

Author(s)

Daniel Hackney, <dan@haxney.org>

See Also

xcmsSource-methods for creating xcmsSource objects in various ways.

xcmsSource-methods

Create an xcmsSource object in a flexible way

Description

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

Methods

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

Author(s)

Daniel Hackney, <dan@haxney.org>

See Also

xcmsSource
XCMSnExp data manipulation methods inherited from MSnbase

Description

The methods listed on this page are XCMSnExp methods inherited from its parent, the OnDiskMSnExp class from the MSnbase package, that alter the raw data or are related to data subsetting. Thus calling any of these methods causes all xcms pre-processing results to be removed from the XCMSnExp object to ensure its data integrity.

The [] method allows to subset a XCMSnExp object by spectra. For more details and examples see the documentation for OnDiskMSnExp.

bin: allows to bin spectra. See bin documentation for more details and examples.
clean: removes unused intensity data points. See clean documentation for details and examples.
filterMsLevel: reduces the XCMSnExp object to spectra of the specified MS level(s). See filterMsLevel documentation for details and examples.
filterAcquisitionNum: filters the XCMSnExp object keeping only spectra with the provided acquisition numbers. See filterAcquisitionNum for details and examples.
The normalize method performs basic normalization of spectra intensities. See normalize documentation for details and examples.
The pickPeaks method performs peak picking. See pickPeaks documentation for details and examples.
The removePeaks method removes mass peaks (intensities) lower than a threshold. Note that these peaks refer to mass peaks, which are different to the chromatographic peaks detected and analyzed in a metabolomics experiment! See removePeaks documentation for details and examples.
The smooth method smooths spectra. See smooth documentation for details and examples.

Usage

```r
## S4 method for signature 'XCMSnExp,logicalOrNumeric,missing,missing'
x[i, j, drop]

## S4 method for signature 'XCMSnExp'
bin(object, binSize = 1L, msLevel.)

clean(object, all = FALSE, verbose = FALSE, msLevel.)

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

## S4 method for signature 'XCMSnExp'
filterAcquisitionNum(object, n, file)

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

## S4 method for signature 'XCMSnExp'
pickPeaks(object, halfWindowSize = 3L, 
    method = c("MAD", "SuperSmoother"), SNR = 0L, ...)

## S4 method for signature 'XCMSnExp'
removePeaks(object, t = "min", verbose = FALSE, 
    msLevel.)

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

Arguments

x For [: an XCMSnExp object.
i For [: numeric or logical vector specifying to which spectra the data set should be reduced.
j For [: not supported.
drop For [: not supported.
object An XCMSnExp or OnDiskMSnExp object.
binSize numeric(1) defining the size of a bin (in Dalton).
msLevel. For bin, clean, filterMsLevel, removePeaks: numeric(1) defining the MS level(s) to which operations should be applied or to which the object should be subsetted.
all For clean: logical(1), if TRUE all zeros are removed.
verbose logical(1) whether progress information should be displayed.
n For filterAcquisitionNum: integer defining the acquisition numbers of the spectra to which the data set should be subsetted.
file For filterAcquisitionNum: integer defining the file index within the object to subset the object by file.
method For normalize: character(1) specifying the normalization method. See normalize for details. For pickPeaks: character(1) defining the method. See pickPeaks for options. For smooth: character(1) defining the method. See smooth for options and details.
...
Optional additional arguments.
halfWindowSize For pickPeaks and smooth: integer(1) defining the window size for the peak picking. See pickPeaks and smooth for details and options.
SNR For pickPeaks: numeric(1) defining the signal to noise ratio to be considered. See pickPeaks documentation for details.
t For removePeaks: either a numeric(1) or "min" defining the threshold (method) to be used. See removePeaks for details.

Value

For all methods: a XCMSnExp object.

Author(s)

Johannes Rainer
See Also

`XCMSnExp-filter` for methods to filter and subset `XCMSnExp` objects. `XCMSnExp` for base class documentation. `OnDiskMSnExp` for the documentation of the parent class.

Description

Subset an `xcmsRaw` object by scans. The returned `xcmsRaw` object contains values for all scans specified with argument `i`. Note that the `scanrange` slot of the returned `xcmsRaw` will be `c(1, length(object@scantime))` and hence not `range(i)`.

Usage

```r
## S4 method for signature 'xcmsRaw,logicalOrNumeric,missing,missing'

x[i, j, drop]
```

Arguments

- `x` The `xcmsRaw` object that should be sub-setted.
- `i` Integer or logical vector specifying the scans/spectra to which `x` should be sub-setted.
- `j` Not supported.
- `drop` Not supported.

Details

Only subsetting by scan index in increasing order or by a logical vector are supported. If not ordered, argument `i` is sorted automatically. Indices which are larger than the total number of scans are discarded.

Value

The sub-setted `xcmsRaw` object.

Author(s)

Johannes Rainer

See Also

`split.xcmsRaw`
Examples

```r
## Load a test file
file <- system.file('cdf/KO/ko15.CDF', package = "faahKO")
xraw <- xcmsRaw(file)
## The number of scans/spectra:
length(xraw@scantime)

## Subset the object to scans with a scan time from 3500 to 4000.
xsub <- xraw[xraw@scantime >= 3500 & xraw@scantime <= 4000]
## The number of scans:
length(xsub@scantime)
## The number of values of the subset:
length(xsub@env$mz)
```
Index

*Topic classes
  xcmsEIC-class, 166
  xcmsFileSource-class, 167
  xcmsFragments-class, 169
  xcmsPeaks-class, 171
  xcmsRaw-class, 174
  xcmsSet-class, 178
  xcmsSource-class, 181
*Topic file
  calibrate-methods, 21
  diffreport-methods, 26
  fillPeaks-methods, 53
  fillPeaks.chrom-methods, 54
  fillPeaks.MSW-methods, 55
  getEIC-methods, 99
  getXcmsRaw-methods, 102
  group.density, 103
  group.mzClust, 104
  group.nearest, 105
  groupnames-methods, 116
  peakTable-methods, 132
  retcor.peakgroups-methods, 150
  sampnames-methods, 152
  verify.mzQuantM, 162
  write.cdf-methods, 163
  write.mzdata-methods, 163
  write.mzQuantML-methods, 164
  writeMzTab, 165
  xcmsFileSource-class, 167
  xcmsFragments, 168
  xcmsRaw, 172
  xcmsSet, 176

*Topic hplot
  image-methods, 118
  levelPlot-methods, 120
  plot.xcmsEIC, 133
  plot.xcmsPeaks, 137
  plotRaw-methods, 138
  plotrt-methods, 139
  plotScan-methods, 140
  plotSpec-methods, 140
  plotSurf-methods, 141
  plotTIC-methods, 141

*Topic iplot
  plotChrom-methods, 135
  plotSpec-methods, 140
  plotSurf-methods, 141
  plotTIC-methods, 141

*Topic lockmass
  AutoLockMass-methods, 14

*Topic manip
  AutoLockMass-methods, 14
  c-methods, 20
  getPeaks-methods, 100
  getScan-methods, 101
  getSpec-methods, 101
  groupval-methods, 117
  medianFilter, 122
  msn2xcmsRaw, 130
  profMedFilt-methods, 145
  profMethod-methods, 145
  profRange-methods, 146
  profStep-methods, 147
  retexp, 151
  specNoise, 156
  specPeaks, 157
  split.xcmsRaw, 158
  split.xcmsSet, 158
  stitch-methods, 160

*Topic methods
  absent-methods, 5
  AutoLockMass-methods, 14
  calibrate-methods, 21
  collect-methods, 25
  diffreport-methods, 26
  fillPeaks-methods, 53
  fillPeaks.chrom-methods, 54
  fillPeaks.MSW-methods, 55
  findMZ, 77
  findneutral, 78
  findPeaks-methods, 80
  findPeaks.addPredictedIsotopeFeatures-methods, 85
  findPeaks.centWave-methods, 87
  findPeaks.centWaveWithPredictedIsotopeROIs-methods,
### INDEX

- **findPeaks.massifquant-methods**, 91
- **findPeaks.MS1-methods**, 95
- **getEIC-methods**, 99
- **getPeaks-methods**, 100
- **getScan-methods**, 101
- **getSpec-methods**, 101
- **getXcmsRaw-methods**, 102
- **group-methods**, 103
- **group.density**, 103
- **group.mzClust**, 104
- **group.nearest**, 105
- **groupnames-methods**, 116
- **groupval-methods**, 117
- **loadRaw-methods**, 121
- **peakPlots-methods**, 131
- **peakTable-methods**, 132
- **plot.xcmsEIC**, 133
- **plotChrom-methods**, 135
- **plotEIC-methods**, 136
- **plotPeaks-methods**, 137
- **plotRaw-methods**, 138
- **plotrt-methods**, 139
- **plotScan-methods**, 140
- **plotSpec-methods**, 140
- **plotSurf-methods**, 141
- **plotTIC-methods**, 141
- **profMedFilt-methods**, 145
- **profMethod-methods**, 145
- **profRange-methods**, 146
- **profStep-methods**, 147
- **rawEIC-methods**, 147
- **rawMat-methods**, 148
- **retcor-methods**, 149
- **retcor.obiwrap**, 149
- **retcor.peakgroups-methods**, 150
- **sampnames-methods**, 152
- **specDist-methods**, 153
- **specDist.cosine**, 154
- **specDist.meanMzmatch**, 155
- **specDist.peakCount-methods**, 155
- **stitch-methods**, 160
- **write.cdf-methods**, 163
- **write.mzdata-methods**, 163
- **write.mzQuantML-methods**, 164
- **xcmsSource-methods**, 181

*Topic models*

- **etg**, 46

*Topic nonlinear*

- **SSgauss**, 159

### Searchable Text

- `findPeaks.massifquant`, 91
- `findPeaks.MS1`, 95
- `getEIC`, 99
- `getPeaks`, 100
- `getScan`, 101
- `getSpec`, 101
- `getXcmsRaw`, 102
- `group`, 103
- `group.density`, 103
- `group.mzClust`, 104
- `group.nearest`, 105
- `groupnames`, 116
- `groupval`, 117
- `loadRaw`, 121
- `peakPlots`, 131
- `peakTable`, 132
- `plot.xcmsEIC`, 133
- `plotChrom`, 135
- `plotEIC`, 136
- `plotPeaks`, 137
- `plotRaw`, 138
- `plotrt`, 139
- `plotScan`, 140
- `plotSpec`, 140
- `plotSurf`, 141
- `plotTIC`, 141
- `profMedFilt`, 145
- `profMethod`, 145
- `profRange`, 146
- `profStep`, 147
- `rawEIC`, 147
- `rawMat`, 148
- `retcor`, 149
- `retcor.obiwrap`, 149
- `retcor.peakgroups`, 150
- `sampnames`, 152
- `specDist`, 153
- `specDist.cosine`, 154
- `specDist.meanMzmatch`, 155
- `specDist.peakCount`, 155
- `stitch`, 160
- `write.cdf`, 163
- `write.mzdata`, 163
- `write.mzQuantML`, 164
- `xcmsSource`, 181

- `findPeaks.MS1-methods`, 95
- `getScan-methods`, 101
- `getPeaks-methods`, 100
- `getSpec-methods`, 101
- `getXcmsRaw-methods`, 102
- `group-methods`, 103
- `group.density`, 103
- `group.mzClust`, 104
- `group.nearest`, 105
- `groupnames-methods`, 116
- `groupval-methods`, 117
- `loadRaw-methods`, 121
- `peakPlots-methods`, 131
- `peakTable-methods`, 132
- `plot.xcmsEIC`, 133
- `plotChrom-methods`, 135
- `plotEIC-methods`, 136
- `plotPeaks-methods`, 137
- `plotRaw-methods`, 138
- `plotrt-methods`, 139
- `plotScan-methods`, 140
- `plotSpec-methods`, 140
- `plotSurf-methods`, 141
- `plotTIC-methods`, 141
- `profMedFilt-methods`, 145
- `profMethod-methods`, 145
- `profRange-methods`, 146
- `profStep-methods`, 147
- `rawEIC-methods`, 147
- `rawMat-methods`, 148
- `retcor-methods`, 149
- `retcor.obiwrap`, 149
- `retcor.peakgroups-methods`, 150
- `sampnames-methods`, 152
- `specDist-methods`, 153
- `specDist.cosine`, 154
- `specDist.meanMzmatch`, 155
- `specDist.peakCount-methods`, 155
- `stitch-methods`, 160
- `write.cdf-methods`, 163
- `write.mzdata-methods`, 163
- `write.mzQuantML-methods`, 164
- `xcmsSource-methods`, 181

*Searchable Text for Models*

- `etg`, 46

*Searchable Text for Nonlinear Models*

- `SSgauss`, 159

### Additional Information

- `[,xcmsRaw,logicalOrNumeric,missing,missing-method`, 184
- `[,xcmsSet,ANY,ANY,ANY-method`, 178
- `[,xcmsSet-method(xcmsSet-class), 178
- $<-,xcmsSet-method(xcmsSet-class), 178
- `absrent(absrent-methods), 5
- `absrent,xcmsSet-method(absrent-methods), 5
- `absrent-methods, 5
- `absMz`, 111
- `absMz,MzClustParam-method`, 111
- `absMz,NearestPeaksParam-method`, 113
- `absMz<-,MzClustParam-method`, 111
- `absMz<-,NearestPeaksParam-method`, 113
- `absRt`, 113
- `absRt,NearestPeaksParam-method`, 113
- `absRt<-,NearestPeaksParam-method`, 113
- `addParams`, 81
- `addParams,MSWParam-method`, 81
- `adjustRtime`, 8, 12
- `adjustRtime(MsFeatureData-class), 122
- `adjustRtime,MsFeatureData-method`, 122
- `adjustedRtime,XCMSnExp-method`, 122
- `adjustedRtime<-,MsFeatureData-class), 122
- `adjustedRtime<-,MsFeatureData-method`, 122
- `adjustedRtime<-,XCMSnExp-method`, 122
- `adjustedRtime<-,XCMSnExp,XCMSnExp-method`, 122
- `adjustedRtime<-,XCMSnExp,ObiwarpParam-method`, 6
- `adjustedRtime<-,XCMSnExp,PeakGroupsParam-method`, 10
INDEX

adjustRtime-obiwarp, 6
adjustRtime-peakGroups, 10
adjustRtimePeakGroups, 12
adjustRtimePeakGroups (adjustRtime-peakGroups), 10
aggregationFun (Chromatogram-class), 22
aggregationFun, Chromatogram-method (Chromatogram-class), 22
ampTh (findPeaks-MSW), 81
ampTh, MSWParam-method (findPeaks-MSW), 81
ampTh<-, MSWParam-method (findPeaks-MSW), 81
array, 172
as.data.frame, Chromatogram-method (Chromatogram-class), 22
AutoLockMass (AutoLockMass-methods), 14
AutoLockMass, xcmsRaw-method (AutoLockMass-methods), 14
AutoLockMass-methods, 14
baseValue (findChromPeaks-matchedFilter), 72
baseValue, MatchedFilterParam-method (findChromPeaks-matchedFilter), 72
baseValue<-, (findChromPeaks-matchedFilter), 72
baseValue<-, MatchedFilterParam-method (findChromPeaks-matchedFilter), 72
bin, 182
bin, XCMSnExp-method ([, XCMSnExp, logicalOrNumeric, missing, method), 182
binSize (findChromPeaks-matchedFilter), 72
binSize, MatchedFilterParam-method (findChromPeaks-matchedFilter), 72
binSize, ObiwarpParam-method (adjustRtime-obiwarp), 6
binSize, PeakDensityParam-method (groupChromPeaks-density), 107
binSize<-, (findChromPeaks-matchedFilter), 72
binSize<-, MatchedFilterParam-method (findChromPeaks-matchedFilter), 72
binSize<-, ObiwarpParam-method (adjustRtime-obiwarp), 6
binSize<-, PeakDensityParam-method (groupChromPeaks-density), 107
binYonX, 15, 19, 20, 39, 40, 76, 144, 166
bplapply, 166, 170
bpparam, 54, 61, 65, 71, 75, 83, 127
breaks_on_binSize, 16, 18, 20
breaks_on_nBins, 16, 19, 19
bw (groupChromPeaks-density), 107
bw, PeakDensityParam-method (groupChromPeaks-density), 107
bw<-, (groupChromPeaks-density), 107
bw<-, PeakDensityParam-method (groupChromPeaks-density), 107
c, 179
c, c-methods (c-methods), 20
c-methods, 20
c.xcmsSet (c-methods), 20
calibrate (calibrate-methods), 21
calibrate, xcmsSet-method (calibrate-methods), 21
CentWaveParam (findChromPeaks-centWave), 58
CentWaveParam, 63, 66, 98, 127, 143
CentWaveParam (findChromPeaks-centWave), 58
CentWaveParam-class (findChromPeaks-centWave), 58
CentWavePredIsoParam, 127
CentWavePredIsoParam (findChromPeaks-centWaveWithPredIsoROIs), 63
CentWavePredIsoParam-class (findChromPeaks-centWaveWithPredIsoROIs), 63
centWaveWithPredIsoROIs, 24
centWaveWithPredIsoROIs (findChromPeaks-centWaveWithPredIsoROIs), 63
character, 167
checkBack (findChromPeaks-massifquant), 67
checkBack, MassifquantParam-method (findChromPeaks-massifquant),
checkBack<-(findChromPeaks-massifquant), 67
checkBack<-,MassifquantParam-method
(findChromPeaks-massifquant), 67
Chromatogram, 47, 48
Chromatogram (Chromatogram-class), 22
Chromatogram-class, 22
chromatographic-peak-detection, 24
chromPeaks, 9, 12, 48, 50
chromPeaks (MsFeatureData-class), 122
chromPeaks,MsFeatureData-method
(MsFeatureData-class), 122
criticalValue,MassifquantParam-method
(findChromPeaks-massifquant), 67
depcopy (xcmsRaw), 172
depcopy,xcmsRaw-method (xcmsRaw), 172
density, 42, 104, 109
diffreport, 5, 132, 179
diffreport (diffreport-methods), 26
diffreport,xcmsSet-method
(diffreport-methods), 26
distance
(findChromPeaks-matchedFilter), 72
distance,MatchedFilterParam-method
(findChromPeaks-matchedFilter), 72
distance<-(findChromPeaks-matchedFilter), 72
distFun (adjustRtime-obiwarp), 6
distFun,ObiwarpParam-method
(adjustRtime-obiwarp), 6
distFun<-(adjustRtime-obiwarp), 6
distFun<-,ObiwarpParam-method
(adjustRtime-obiwarp), 6
do_adjustRtime_peakGroups, 13, 28
do_adjustRtime_peakGroups, 13, 28
do_findChromPeaks_massifquant
(findChromPeaks-massifquant), 32
do_findChromPeaks_centWaveWithPredIsoROIs
(do_findChromPeaks_centWaveWithPredIsoROIs), 32
do_findChromPeaks_centWave, 29, 35, 37,
40, 41, 62, 69
do_findChromPeaks_centWaveWithPredIsoROIs,
32, 35, 40, 41, 66, 91
do_findChromPeaks_massifquant, 32, 35,
35, 40, 41, 72
do_findChromPeaks_matchedFilter, 32, 35,
37, 38, 41, 76, 94, 95, 162
do_findPeaks_MSW, 32, 35, 37, 40, 40, 84, 98
do_groupChromPeaks_density, 42, 45, 46,
104, 110
do_groupChromPeaks_nearest, 43, 44, 46,
115
do_groupPeaks_mzClust, 43, 45, 45, 113
dropAdjustedRtime
(MsFeatureData-class), 122
dropAdjustedRtime, MsFeatureData-method (MsFeatureData-class), 122
dropAdjustedRtime, XCMSnExp-method (MsFeatureData-class), 122
dropChromPeaks (MsFeatureData-class), 122
dropChromPeaks, MsFeatureData-method (MsFeatureData-class), 122
dropChromPeaks, XCMSnExp-method (MsFeatureData-class), 122
dropFeatureDefinitions (MsFeatureData-class), 122
dropFeatureDefinitions, MsFeatureData-method (MsFeatureData-class), 122
dropFeatureDefinitions, XCMSnExp-method (MsFeatureData-class), 122
dropFilledChromPeaks, 52
dropFilledChromPeaks (MsFeatureData-class), 122
dropFilledChromPeaks, XCMSnExp-method (MsFeatureData-class), 122
dropFilledChromPeaks<-, 52
etg, 46
expandMz (FillChromPeaksParam-class), 49
expandMz, FillChromPeaksParam-method (FillChromPeaksParam-class), 49
expandMz<-, FillChromPeaksParam-method (FillChromPeaksParam-class), 49
expandMz<-, FillChromPeaksParam-method (FillChromPeaksParam-class), 49
expandRt (FillChromPeaksParam-class), 49
expandRt, FillChromPeaksParam-method (FillChromPeaksParam-class), 49
expandRt<-, FillChromPeaksParam-method (FillChromPeaksParam-class), 49
expandRt<-, FillChromPeaksParam-method (FillChromPeaksParam-class), 49
extrChromatograms, 22, 24
extrChromatograms (extrChromatograms, OnDiskMSnExp-method), 47
extrChromatograms, OnDiskMSnExp-method, 47
extrChromatograms, XCMSnExp-method (extrChromatograms, OnDiskMSnExp-method), 47
extrChromatograms<-, 51
extrChromatograms<-, OnDiskMSnExp-method, 47
extrChromatograms<-, XCMSnExp-method (extrChromatograms, OnDiskMSnExp-method), 47
extraPeaks (adjustRtime-peakGroups), 10
extraPeaks, PeakGroupsParam-method (adjustRtime-peakGroups), 10
extraPeaks<-, adjustRtime-peakGroups), 10
extraPeaks<-, PeakGroupsParam-method (adjustRtime-peakGroups), 10
factorDiag (adjustRtime-obiwarp), 6
factorDiag, ObiwarpParam-method (adjustRtime-obiwarp), 6
factorDiag<-, adjustRtime-obiwarp), 6
factorDiag<-, ObiwarpParam-method (adjustRtime-obiwarp), 6
factorGap (adjustRtime-obiwarp), 6
factorGap, ObiwarpParam-method (adjustRtime-obiwarp), 6
family (adjustRtime-peakGroups), 10
family, PeakGroupsParam-method (adjustRtime-peakGroups), 10
family<-, adjustRtime-peakGroups), 10
family<-, PeakGroupsParam-method (adjustRtime-peakGroups), 10
featureDefinitions, 49, 107, 109, 110, 112, 113, 115, 129
featureDefinitions (MsFeatureData-class), 122
featureDefinitions, MsFeatureData-method (MsFeatureData-class), 122
featureDefinitions, XCMSnExp-method (MsFeatureData-class), 122
featureDefinitions<-, (MsFeatureData-class), 122
featureDefinitions<-, MsFeatureData-method (MsFeatureData-class), 122
featureDefinitions<-, XCMSnExp-method (MsFeatureData-class), 122
featureValues, 51
featureValues (featureValues, XCMSnExp-method), 48
featureValues, XCMSnExp-method, 48
fileIndex (ProcessHistory-class), 142
fileIndex, ProcessHistory-method (ProcessHistory-class), 142
filepaths (xcmsSet-class), 178
filepaths, xcmsSet-method (xcmsSet-class), 178
filepaths<-, xcmsSet-method (xcmsSet-class), 178
fillChromPeaks, 49, 124, 129
fillChromPeaks (fillChromPeaksParam-class), 49
fillChromPeaks, XCMSnExp, FillChromPeaksParam-method (fillChromPeaksParam-class), 49
(findPeaks.massifquant-methods), 91
findPeaks.massifquant, xcmsRaw-method (findPeaks.massifquant-methods), 91
findPeaks.massifquant-methods, 91
findPeaks.matchedFilter, 76, 80, 177, 178
findPeaks.matchedFilter (findPeaks.matchedFilter, xcmsRaw-method), 94
findPeaks.matchedFilter, xcmsRaw-method, 94
findPeaks.MS1 (findPeaks.MS1-methods), 95
findPeaks.MS1, xcmsRaw-method (findPeaks.MS1-methods), 95
findPeaks.MSW (findPeaks.MSW-methods), 97
findPeaks.MSW, xcmsRaw-method, 97
firstBaselineCheck (findChromPeaks-centWave), 58
firstBaselineCheck, CentWaveParam-method (findChromPeaks-centWave), 58
firstBaselineCheck<- (findChromPeaks-centWave), 58
firstBaselineCheck<-, CentWaveParam-method (findChromPeaks-centWave), 58
fitgauss (findChromPeaks-centWave), 58
fitgauss, CentWaveParam-method (findChromPeaks-centWave), 58
fitgauss, MassifquantParam-method (findChromPeaks-massifquant), 67
fitgauss<- (findChromPeaks-centWave), 58
fitgauss<-, CentWaveParam-method (findChromPeaks-centWave), 58
fitgauss<-, MassifquantParam-method (findChromPeaks-massifquant), 67
fromFile, Chromatogram-method (Chromatogram-class), 22
fwhm (findChromPeaks-matchedFilter), 72
fwhm, MatchedFilterParam-method (findChromPeaks-matchedFilter), 72
fwhm<- (findChromPeaks-matchedFilter), 72
fwhm<-, MatchedFilterParam-method (findChromPeaks-matchedFilter), 72
gapExtend (adjustRtime-obiwarp), 6
gapExtend, ObiwarpParam-method (adjustRtime-obiwarp), 6
gapExtend<- (adjustRtime-obiwarp), 6
gapExtend<-, ObiwarpParam-method (adjustRtime-obiwarp), 6
gapInit (adjustRtime-obiwarp), 6
gapInit, ObiwarpParam-method (adjustRtime-obiwarp), 6
gapInit<-, ObiwarpParam-method (adjustRtime-obiwarp), 6
GenericParam (GenericParam-class), 98
getEIC, 147, 152, 166, 167, 175, 179
getEIC (getEIC-methods), 99
getEIC, xcmsRaw-method (getEIC-methods), 99
group, 5, 13, 107, 109, 112, 115, 178, 180
group (group-methods), 103
group, xcmsSet-method (group-methods), 103
group-methods, 103
group.density, 103, 103, 106, 110
<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>group.density.xcmsSet-method</td>
<td>103</td>
</tr>
<tr>
<td>group.mzClust.xcmsSet-method</td>
<td>104, 106, 113</td>
</tr>
<tr>
<td>group.nearest.xcmsSet-method</td>
<td>105, 115</td>
</tr>
<tr>
<td>groupChromPeaks.xCMSnExp.mzClustParam-method</td>
<td>111</td>
</tr>
<tr>
<td>groupChromPeaks.xCMSnExp.NearestPeaksParam-method</td>
<td>113</td>
</tr>
<tr>
<td>groupChromPeaks.xCMSnExp.PeakDensityParam-method</td>
<td>107</td>
</tr>
<tr>
<td>groupChromPeaks-density</td>
<td>107</td>
</tr>
<tr>
<td>groupChromPeaks-mzClust</td>
<td>111</td>
</tr>
<tr>
<td>groupChromPeaks-nearest</td>
<td>113</td>
</tr>
<tr>
<td>groupidx.xcmsSet-method</td>
<td>178</td>
</tr>
<tr>
<td>index.xcmsSet-method</td>
<td>72</td>
</tr>
<tr>
<td>groupnames.xcmsEIC-method</td>
<td>116</td>
</tr>
<tr>
<td>groupnames.xcmsSet-method</td>
<td>116</td>
</tr>
<tr>
<td>groups.xcmsSet-method</td>
<td>178</td>
</tr>
<tr>
<td>initPenalty.adjustRtime.obiwarp-obiwarp</td>
<td>6</td>
</tr>
<tr>
<td>initPenalty.obiwarpParam-method</td>
<td>6</td>
</tr>
<tr>
<td>groupval.xcmsSet-method</td>
<td>178</td>
</tr>
<tr>
<td>groupval.xCMSnExp.massifquantParam-method</td>
<td>58</td>
</tr>
<tr>
<td>hasAdjustedRtime.MsFeatureData-class</td>
<td>122</td>
</tr>
<tr>
<td>hasAdjustedRtime.MsFeatureData-method</td>
<td>122</td>
</tr>
<tr>
<td>hasChromPeaks.MsFeatureData-class</td>
<td>122</td>
</tr>
<tr>
<td>hasChromPeaks.MsFeatureData-method</td>
<td>122</td>
</tr>
<tr>
<td>hasChromPeaks.xCMSnExp-method</td>
<td>122</td>
</tr>
<tr>
<td>hasFeatures.MsFeatureData-class</td>
<td>122</td>
</tr>
<tr>
<td>hasFeatures.xCMSnExp-method</td>
<td>122</td>
</tr>
<tr>
<td>identifyMajorPeaks</td>
<td>81, 83, 97</td>
</tr>
<tr>
<td>image.xcmsRaw-method</td>
<td>118</td>
</tr>
<tr>
<td>image.xcmsSet-method</td>
<td>118</td>
</tr>
<tr>
<td>impute.matchedFilterParam-method</td>
<td>72</td>
</tr>
<tr>
<td>impute.linInterpol</td>
<td>118, 126, 127</td>
</tr>
<tr>
<td>index.xCMSnExp.massifquantParam-method</td>
<td>58</td>
</tr>
<tr>
<td>integrate.CentWaveParam-method</td>
<td>58</td>
</tr>
<tr>
<td>integrate.massifquantParam-method</td>
<td>67</td>
</tr>
<tr>
<td>integrate&lt;-xCMSnExp.massifquantParam-method</td>
<td>58</td>
</tr>
<tr>
<td>integrate&lt;-CentWaveParam-method</td>
<td>58</td>
</tr>
<tr>
<td>integrate&lt;-MassifquantParam-method</td>
<td>67</td>
</tr>
</tbody>
</table>
intensity, Chromatogram-method (Chromatogram-class), 22
intensity, XCMSnExp-method (MsFeatureData-class), 122
kNN (groupChromPeaks-nearest), 113
kNN, NearestPeaksParam-method (groupChromPeaks-nearest), 113
kNN<-, NearestPeaksParam-method (groupChromPeaks-nearest), 113
length, Chromatogram-method (Chromatogram-class), 22
levelplot, 175
levelplot(xcmsRaw-class), 174
levelplot, xcmsRaw-method (levelplot-methods), 120
levelplot, xcmsSet-method (levelplot-methods), 120
loadRaw (loadRaw-methods), 121
loadRaw, xcmsfileSource-method (loadRaw-methods), 121
loadRaw, xcmsSource-method (loadRaw-methods), 121
loadRaw-methods, 121
localAlignment (adjustRtime-obiwarp), 6
localAlignment, ObiwarpParam-method (adjustRtime-obiwarp), 6
localAlignment<-(adjustRtime-obiwarp), 6
localAlignment<-, ObiwarpParam-method (adjustRtime-obiwarp), 6
loess, 12, 28, 151
makeacqNum (stitch-methods), 160
makeacqNum, xcmsRaw-method (stitch-methods), 160
massifquant, 24, 37
massifquant (findChromPeaks-massifquant), 67
MassifquantParam, 127
MassifquantParam (findChromPeaks-massifquant), 67
MassifquantParam-class (findChromPeaks-massifquant), 67
matchedFilter, 24, 40, 51, 95
matchedFilter (findChromPeaks-matchedFilter), 72
MatchedFilterParam, 127
MatchedFilterParam (findChromPeaks-matchedFilter), 72
MatchedFilterParam-class (findChromPeaks-matchedFilter), 72
matrix, 172
max, MatchedFilterParam-method (findChromPeaks-matchedFilter), 72
max<-(findChromPeaks-matchedFilter), 72
max<-, MatchedFilterParam-method (findChromPeaks-matchedFilter), 72
maxCharge (findChromPeaks-centWaveWithPredIsoROIs), 63
maxCharge, CentWavePredIsoParam-method (findChromPeaks-centWaveWithPredIsoROIs), 63
maxCharge<-(findChromPeaks-centWaveWithPredIsoROIs), 63
maxCharge<-, CentWavePredIsoParam-method (findChromPeaks-centWaveWithPredIsoROIs), 63
maxFeatures (groupChromPeaks-density), 107
maxFeatures, PeakDensityParam-method (groupChromPeaks-density), 107
maxFeatures<-(groupChromPeaks-density), 107
maxFeatures<-, PeakDensityParam-method (groupChromPeaks-density), 107
maxIso (findChromPeaks-centWaveWithPredIsoROIs), 63
maxIso, CentWavePredIsoParam-method (findChromPeaks-centWaveWithPredIsoROIs), 63
maxIso<-(findChromPeaks-centWaveWithPredIsoROIs), 63
maxIso<-, CentWavePredIsoParam-method (findChromPeaks-centWaveWithPredIsoROIs), 63
medianfilter, 122, 145
minFraction (groupChromPeaks-density), 107
minFraction, MzClustParam-method (groupChromPeaks-mzClust), 111
minFraction, PeakDensityParam-method
(groupChromPeaks-density, 107)
minFraction, PeakGroupsParam-method
(adjustRtime-peakGroups), 10
minFraction<-
(groupChromPeaks-density, 107)
minFraction<-, MzClustParam-method
(groupChromPeaks-mzClust), 111
minFraction<-, PeakDensityParam-method
(groupChromPeaks-density), 107
minFraction<-, PeakGroupsParam-method
(adjustRtime-peakGroups), 10
minNoiseLevel (findPeaks-MSW), 81
minNoiseLevel, MSWParam-method
(findPeaks-MSW), 81
minNoiseLevel<-(findPeaks-MSW), 81
minSamples (groupChromPeaks-density), 107
minSamples, MzClustParam-method
(groupChromPeaks-mzClust), 111
minSamples, PeakDensityParam-method
(groupChromPeaks-density), 107
minSamples<-(groupChromPeaks-density), 107
minSamples<-, MzClustParam-method
(groupChromPeaks-mzClust), 111
minSamples<-, PeakDensityParam-method
(groupChromPeaks-density), 107
MsFeatureData (MsFeatureData-class), 122
MsFeatureData-class, 122
mzlevel (xcmsSet-class), 178
mslevel, xcmsRaw-method (xcmsRaw-class), 174
mslevel, xcmsSet-method (xcmsSet-class), 178
msn2xcmsRaw, 130
MSnExp, 62, 66, 71, 76, 84, 129
MSW, 24, 41, 51, 98
MSW (findPeaks-MSW), 81
MSWParam, 127
MSWParam (findPeaks-MSW), 81
MSWParam-class (findPeaks-MSW), 81
mt.teststat, 26, 27
MulticoreParam, 177
mz, Chromatogram-method
(Chromatogram-class), 22
mz, XCMSnExp-method
(MsFeatureData-class), 122
mzCenterFun (findChromPeaks-centWave), 58
mzCenterFun, CentWaveParam-method
(findChromPeaks-centWave), 58
mzCenterFun, MassifquantParam-method
(findChromPeaks-massifquant), 67
mzCenterFun<-(findChromPeaks-centWave), 58
mzCenterFun<-, MassifquantParam-method
(findChromPeaks-massifquant), 67
MzClustParam (groupChromPeaks-mzClust), 111
MzClustParam-class
(groupChromPeaks-mzClust), 111
mzdiff (findChromPeaks-centWave), 58
mzdiff, CentWaveParam-method
(findChromPeaks-centWave), 58
mzdiff, MassifquantParam-method
(findChromPeaks-massifquant), 67
mzdiff, MatchedFilterParam-method
(findChromPeaks-matchedFilter), 72
mzdiff<-(findChromPeaks-centWave), 58
mzdiff<-, CentWaveParam-method
(findChromPeaks-centWave), 58
mzdiff<-, MassifquantParam-method
(findChromPeaks-massifquant), 67
mzdiff<-, MatchedFilterParam-method
(findChromPeaks-matchedFilter), 72
mzIntervalExtension
(findChromPeaks-centWaveWithPredIsoROIs), 63
mzIntervalExtension, CentWavePredIsoParam-method
(findChromPeaks-centWaveWithPredIsoROIs), 63
mzIntervalExtension<-(findChromPeaks-centWaveWithPredIsoROIs), 63
mzIntervalExtension<-, CentWavePredIsoParam-method
(findChromPeaks-centWaveWithPredIsoROIs), 63
mzrange (xcmsEIC-class), 166
mzrange, xcmsEIC-method (xcmsEIC-class), 166
mzVsRtBalance
(groupChromPeaks-nearest), 113
mzVsRtBalance, NearestPeaksParam-method
INDEX

(xcmsSet-class), 178
phenoData<-,xcmsSet-method (xcmsSet-class), 178
pickPeaks, 182, 183
pickPeaks,XCMSnExp-method ([,XCMSnExp,logicalOrNumeric,missing,missing-method, xcmsSet-class], 178
plot, 166
plot, plot-methods (plot.xcmsEIC), 133
plot.xcmsEIC, 133
plotAdjustedRtime, 6, 9, 13, 134
plotChrom, 136, 145, 175
plotChrom (plotChrom-methods), 135
plotChrom, xcmsRaw-method (plotChrom-methods), 135
plotChrom-methods, 135
plotEIC (plotEIC-methods), 136
plotEIC, xcmsRaw-method (plotEIC-methods), 136
plotEIC-methods, 136
plotPeaks (plotPeaks-methods), 137
plotPeaks, xcmsRaw-method (plotPeaks-methods), 137
plotPeaks-methods, 137
plotQC, 137
plotRaw, 148, 175
plotRaw (plotRaw-methods), 138
plotRaw, xcmsRaw-method (plotRaw-methods), 138
plotRaw-methods, 138
plotrt, 180
plotrt (plotrt-methods), 139
plotrt, xcmsSet-method (plotrt-methods), 139
plotrt-methods, 139
plotScan, 175
plotScan (plotScan-methods), 140
plotScan, xcmsRaw-method (plotScan-methods), 140
plotScan-methods, 140
plotSpec, 145, 175
plotSpec (plotSpec-methods), 140
plotSpec, xcmsRaw-method (plotSpec-methods), 140
plotSpec-methods, 140
plotSurf, 175
plotSurf (plotSurf-methods), 141
plotSurf, xcmsRaw-method (plotSurf-methods), 141
plotSurf-methods, 141
plotTIC, 175
plotTIC (plotTIC-methods), 141
plotTIC,xcmsRaw-method (plotTIC-methods), 141
plotTIC-methods, 141
plotTree (xcmsFragments-class), 169
plotTree, xcmsFragments-method (plotTree-methods, xcmsFragments-class), 169
png, 134
polarity, CentWavePredIsoParam-method (findChromPeaks-centWaveWithPredIsoROIs), 63
polarity<-, (findChromPeaks-centWaveWithPredIsoROIs), 63
polarity<-, CentWavePredIsoParam-method (findChromPeaks-centWaveWithPredIsoROIs), 63
postscript, 134
ppm (findChromPeaks-centWave), 58
ppm, CentWaveParam-method (findChromPeaks-centWave), 58
ppm, FillChromPeaksParam-method (fillChromPeaksParam-class), 49
ppm, MassifquantParam-method (findChromPeaks-massifquant), 67
ppm, MzClustParam-method (groupChromPeaks-mzClust), 111
ppm<-, (findChromPeaks-centWave), 58
ppm<-, CentWaveParam-method (findChromPeaks-centWave), 58
ppm<-, FillChromPeaksParam-method (fillChromPeaksParam-class), 49
ppm<-, MassifquantParam-method (findChromPeaks-massifquant), 67
ppm<-, MzClustParam-method (groupChromPeaks-mzClust), 111
ppm<-, CentWaveParam-method (findChromPeaks-centWave), 58
ppm<-, MassifquantParam-method (findChromPeaks-massifquant), 67
precursorMz, Chromatogram-method (Chromatogram-class), 22
prefilter (findChromPeaks-centWave), 58
prefilter, CentWaveParam-method (findChromPeaks-centWave), 58
prefilter, MassifquantParam-method (findChromPeaks-massifquant), 67
prefilter<-, (findChromPeaks-centWave), 58
prefilter<-, CentWaveParam-method (findChromPeaks-centWave), 58
prefilter<-, MassifquantParam-method (findChromPeaks-massifquant), 67
present (absent-methods), 5
present, xcmsSet-method (absent-methods), 5
processDate (ProcessHistory-class), 142
processDate, ProcessHistory-method (ProcessHistory-class), 142
ProcessHistory, 123, 127, 128
ProcessHistory (ProcessHistory-class), 142
processHistory, 13, 98
processHistory, XMsfeatureData-class), 122
processHistory, XCMSSnExp-method (MsfeatureData-class), 122
ProcessHistory-class, 142
processHistoryTypes, 143
processHistoryTypes (MsfeatureData-class), 122
processInfo (ProcessHistory-class), 142
processInfo, ProcessHistory-method (ProcessHistory-class), 142
processParam (ProcessHistory-class), 142
processParam, XProcessHistory-method (ProcessHistory-class), 142
processType (ProcessHistory-class), 142
processType, ProcessHistory-method (ProcessHistory-class), 142
productMz (Chromatogram-class), 22
productMz, Chromatogram-method (Chromatogram-class), 22
profBin, 145, 177, 178
profBinLin, 177
profBinLinBase, 177
profile-matrix (profMat-xcmsSet), 143
profinfo, 175
profinfo (xcmsSet-class), 178
profinfo, xcmsRaw-method (xcmsRaw-class), 174
profinfo, xcmsSet-method (xcseset-class), 178
profinfo, xcseset-method (xcseset-class), 178
profIntLin, 177
profMat, 123, 174
profMat (profMat-xcseset), 143
profMat, XmsfeatureData-Method, 122
profMat, XCMSSnExp-method (MsfeatureData-class), 122
profMat, XCMSSnExp-method (MsfeatureData-class), 122
profMat, xcmsRaw-method (profMat-xcseset), 143
profMat-xcmsSet, 143
profMedFilt, 175
profMedFilt (profMedFilt-methods), 145
profMedFilt, xcmsRaw-method (profMedFilt-methods), 145
profMedFilt-methods, 145
profMethod, 145, 147, 173, 176, 178
profMethod (profMethod-methods), 145
profMethod, xcmsRaw-method (profMethod-methods), 145
profMethod, xcmsSet-method (xcseset-class), 178
profMethod-methods, 145
profMethod<-, 176
profMethod<-(profMethod-methods), 145
profMethod<-, xcmsRaw-method (profMethod-methods), 145
profMz (xcmsRaw-class), 174
profMz, xcmsRaw-method (xcmsRaw-class), 174
profRange, 101, 102, 136, 140, 141, 176
profRange (profRange-methods), 146
profRange, xcmsRaw-method (profRange-methods), 146
profRange-methods, 146
profStep, 173, 176, 178
profStep (profStep-methods), 147
profStep, xcmsRaw-method (profStep-methods), 147
profStep, xcmsSet-method (xcseset-class), 178
profStep-methods, 147
profStep<-, 176
profStep<-(profStep-methods), 147
profStep<-, xcmsRaw-method (profStep-methods), 147
progressCallback (xcmsSet-class), 178
progressCallback, xcmsSet-method (xcseset-class), 178
progressCallback<-(xcseset-class), 178
progressCallback<-(xcseset-method (xcseset-class), 178
pSet, 129
rawEIC, 99, 136
rawEIC (rawEIC-methods), 147
rawEIC, xcmsRaw-method (rawEIC-methods), 147
rawEIC-methods, 147
rawMat (rawMat-methods), 148
rawMat, xcmsRaw-method (rawMat-methods), 148
rawMat-methods, 148
register, 61, 66, 71, 76, 83
removePeaks, 182, 183
removePeaks, XCMSnExp-method
   ([, XCMSnExp, logicalOrNumeric, missing, ]),
   182
response (adjustRtime-obiwarp), 6
response, ObiwapParam-method
   (adjustRtime-obiwarp), 6
response<-, ObiwapParam-method
   (adjustRtime-obiwarp), 6
retcor, 6, 9, 139, 180
retcor (retcor-methods), 149
retcor, XCMSnExp-method
   (retcor-methods), 149
retcor-methods, 149
retcor.linear
   (retcor.peakgroups-methods), 150
retcor.linear, XCMSnExp-method
   (retcor.peakgroups-methods), 150
retcor.loess, 149
retcor.loess
   (retcor.peakgroups-methods), 150
retcor.loess, XCMSnExp-method
   (retcor.peakgroups-methods), 150
retcor.obiwarp, 9, 149, 149, 151
retcor.obiwarp, XCMSnExp-method
   (retcor.obiwarp), 149
retcor.peakgroups, 13
retcor.peakgroups
   (retcor.peakgroups-methods), 150
retcor.peakgroups, XCMSnExp-method
   (retcor.peakgroups-methods), 150
retcor.peakgroups-methods, 150
retexp, 151
revMz (xcmsRaw-class), 174
revMz, XCMSnExp-method
   (xcmsRaw-class), 174
ridgeLength (findPeaks-MSW), 81
ridgeLength, MSWPam-method
   (findPeaks-MSW), 81
ridgeLength<-, (findPeaks-MSW), 81
ridgeLength<-, MSWPam-method
   (findPeaks-MSW), 81
roiList (findChromPeaks-centWave), 58
roiList<-, (findChromPeaks-centWave), 58
roiList<-, CentWaveParam-method
   (findChromPeaks-centWave), 58
roiScales (findChromPeaks-centWave), 58
roiScales, CentWaveParam-method
   (findChromPeaks-centWave), 58
roiScales<-, (findChromPeaks-centWave), 58
roiScales<-, CentWaveParam-method
   (findChromPeaks-centWave), 58
rtime, Chromatogram-method
   (Chromatogram-class), 22
rtime, XCMSnExp-method
   (MsFeatureData-class), 122
rtrange (xcmsEIC-class), 166
rtrange, XCMSnExp-method
   (xcmsEIC-class), 166
sampclass, 5, 177
sampclass (xcmsSet-class), 178
sampclass, XCMSnExp-method
   (xcmsSet-class), 178
sampclass<-, (xcmsSet-class), 178
sampleGroups (groupChromPeaks-density), 107
sampleGroups, MzClustParam-method
   (groupChromPeaks-mzClust), 111
sampleGroups, NearestPeaksParam-method
   (groupChromPeaks-nearest), 113
sampleGroups, PeakDensityParam-method
   (groupChromPeaks-density), 107
sampleGroups<-
   (groupChromPeaks-density), 107
sampleGroups<-, MzClustParam-method
   (groupChromPeaks-mzClust), 111
sampleGroups<-, NearestPeaksParam-method
   (groupChromPeaks-nearest), 113
sampleGroups<-, PeakDensityParam-method
   (groupChromPeaks-density), 107
sampnames, 166, 180
sampnames (sampnames-methods), 152
sampnames, XCMSnExp-method
   (sampnames-methods), 152
sampnames, XCMSnSet-method
   (sampnames-methods), 152
sampnames-methods, 152
sampnames<-, (xcmsSet-class), 178
sampnames<-, XCMSnSet-method
   (xcmsSet-class), 178
sav.gol, 81, 83, 97
(findChromPeaks-centWaveWithPredIsoROIs), 72
steps<-(findChromPeaks-matchedFilter),
63
snthreshIsoROIs,CentWavePredIsoParam-method
(findChromPeaks-centWaveWithPredIsoROIs)>
63
snthreshIsoROIs<-
(findChromPeaks-centWaveWithPredIsoROIs)>
63
snthreshIsoROIs<-,CentWavePredIsoParam-method
(findChromPeaks-centWaveWithPredIsoROIs)
63

sortMz (xcmsRaw-class), 174
sortMz, xcmsRaw-method (xcmsRaw-class),
174
span (adjustRtime-peakGroups), 10
span, PeakGroupsParam-method
(adjustRtime-peakGroups), 10
span<-(adjustRtime-peakGroups), 10
span<-, PeakGroupsParam-method
(adjustRtime-peakGroups), 10
specDist (specDist-methods), 153
specDist, xcmsSet-method
(specDist-methods), 153
specDist-methods, 153
specDist.cosine, 154
specDist.cosine,matrix,matrix-method
(specDist.cosine), 154
specDist.meanMZmatch, 155
specDist.meanMZmatch,matrix,matrix-method
(specDist.meanMZmatch), 155
specDist.peakCount
(specDist.peakCount-methods), 155
specDist.peakCount,matrix,matrix-method
(specDist.peakCount-methods), 155
specDist.peakCount-methods, 155
specNoise, 156, 157
specPeaks, 156, 157
spectra, XCMSnExp-method
(MsFeatureData-class), 122
Spectrum, 123, 128
split, 180
split, split-methods (split.xcmsSet),
158
split.screen, 131, 137
split.xcmsRaw, 158, 184
split.xcmsSet, 158
SSgauss, 159
steps (findChromPeaks-matchedFilter), 72
steps, MatchedFilterParam-method
(findChromPeaks-matchedFilter),
63
stitch (stitch-methods), 160
stitch, xcmsRaw-method (stitch-methods),
160
stitch-methods, 160
structure, 172
subset-xcmsRaw
([, xcmsRaw, logicalOrNumeric, missing, missing-method),
184
tuneIn (findPeaks-MSW), 81
tuneIn, MSWParam-method (findPeaks-MSW),
81
tuneIn<-(findPeaks-MSW), 81
tuneIn<-,MSWParam-method
(findPeaks-MSW), 81
tuneInPeakInfo, 41, 83, 97
unions (findChromPeaks-massifquant), 67
unions, MassifquantParam-method
(findChromPeaks-massifquant),
67
unions<-(findChromPeaks-massifquant), 67
unions<-, MassifquantParam-method
(findChromPeaks-massifquant),
67
updateObject, xcmsSet-method, 161
useOriginalCode, 161
vector, 172
verboseColumns
(findChromPeaks-centWave), 58
verboseColumns, CentWaveParam-method
(findChromPeaks-centWave), 58
verboseColumns, MassifquantParam-method
(findChromPeaks-massifquant),
67
verboseColumns, MSWParam-method
(findPeaks-MSW), 81
verboseColumns<-(findChromPeaks-centWave), 58
verboseColumns<-, CentWaveParam-method
(findChromPeaks-centWave), 58
verboseColumns<-, MassifquantParam-method
(findChromPeaks-massifquant),
67
verboseColumns<-, MSWParam-method
(findPeaks-MSW), 81
verify.mzQuantM, 162
verify.mzQuantM, 164
verify.mzQuantML (verify.mzQuantM), 162
withWave (findChromPeaks-massifquant), 67
withWave, MassifquantParam-method
(findChromPeaks-massifquant), 67
withWave<-,
(findChromPeaks-massifquant), 67
write.cdf (write.cdf-methods), 163
write.cdf, xcmsRaw-method
(write.cdf-methods), 163
write.mzdata (write.mzdata-methods), 163
write.mzdata, xcmsRaw-method
(write.mzdata-methods), 163
write.mzdata-methods, 163
write.mzQuantML, 162
write.mzQuantML
(write.mzQuantML-methods), 164
write.mzQuantML, xcmsSet-method
(write.mzQuantML-methods), 164
write.mzQuantML-methods, 164
writeMzTab, 165
xcmsRaw-class, 174
xcmsSet-class, 178
xcmsSource, 121, 167, 181
xcmsSource (xcmsSource-methods), 181
xcmsSource, character-method
(xcmsFileSource-class), 167
xcmsSource, xcmsSource-method
(xcmsSource-methods), 181
xcmsSource-class, 181
xcmsSource-methods, 181
XProcessHistory (ProcessHistory-class), 142
XProcessHistory-class
(ProcessHistory-class), 142