

# Package ‘flagme’

May 8, 2024

**Version** 1.61.0

**Date** 2015/04/06

**Title** Analysis of Metabolomics GC/MS Data

**Author** Mark Robinson <mark.robinson@imls.uzh.ch>, Riccardo Romoli <riccardo.romoli@unifi.it>

**Maintainer** Mark Robinson <mark.robinson@imls.uzh.ch>, Riccardo Romoli <riccardo.romoli@unifi.it>

**Depends** gcspikelite, xcms, CAMERA

**Imports** gplots, graphics, MASS, methods, SparseM, stats, utils

**Description** Fragment-level analysis of gas chromatography-massspectrometry metabolomics data.

**License** LGPL (>= 2)

**Collate** 0classes.R clusterAlignment.R init.R multipleAlignment.R peaksAlignment.R progressiveAlignment.R betweenAlignment.R dp.R metrics.R parse.R peaksDataset.R gatherInfo.R plotFragments.R rmaFitUnit.R addXCMSPeaks.R retFatMatrix.R exportSpectra.R importSpectra.R

**biocViews** DifferentialExpression, MassSpectrometry

**RoxygenNote** 7.2.3

**Encoding** UTF-8

**git\_url** <https://git.bioconductor.org/packages/flagme>

**git\_branch** devel

**git\_last\_commit** 05d260a

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.20

**Date/Publication** 2024-05-08

## Contents

addAMDISPeaks . . . . .	3
addChromaTOFPeaks . . . . .	4
addXCMSPeaks . . . . .	5
betweenAlignment . . . . .	7
calcTimeDiffs . . . . .	8
clusterAlignment . . . . .	9
compress,peaksAlignment-method . . . . .	11
compress,progressiveAlignment-method . . . . .	12
corPrt . . . . .	12
decompress,peaksAlignment-method . . . . .	13
decompress,progressiveAlignment-method . . . . .	14
deDuper . . . . .	15
distToLib . . . . .	15
dp . . . . .	16
dynRT . . . . .	17
eitherMatrix-class . . . . .	18
exportSpectra . . . . .	18
gatherInfo . . . . .	19
headToTailPlot . . . . .	21
importSpec . . . . .	21
imputePeaks . . . . .	22
matchSpec . . . . .	23
multipleAlignment-class . . . . .	24
ndpRT . . . . .	26
normDotProduct . . . . .	27
parseChromaTOF . . . . .	29
parseELU . . . . .	30
peaksAlignment-class . . . . .	31
peaksDataset . . . . .	34
plotAlignedFragments . . . . .	35
plotAlignment,peaksAlignment-method . . . . .	36
plotChrom,peaksDataset-method . . . . .	38
plotClustAlignment,clusterAlignment-method . . . . .	40
plotFragments . . . . .	41
plotImage . . . . .	42
progressiveAlignment-class . . . . .	44
retFatMatrix . . . . .	45
rmaFitUnit . . . . .	47
show,multipleAlignment-method . . . . .	48

## Index

50

---

addAMDISPeaks	<i>Add AMDIS peak detection results</i>
---------------	---

---

### Description

Reads ASCII ELU-format files (output from AMDIS) and attaches them to an already created peaksDataset object

### Usage

```
addAMDISPeaks(object, fns = dir(, "[Eu][Ll][Uu]"), verbose = TRUE, ...)
```

### Arguments

object	a peaksDataset object.
fns	character vector of same length as object@rawdata (user ensures the order matches)
verbose	whether to give verbose output, default TRUE
...	arguments passed on to parseELU

### Details

Repeated calls to parseELU to add peak detection results to the original peaksDataset object.

### Value

peaksDataset object

### Author(s)

Mark Robinson

### References

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

### See Also

[parseELU](#), [peaksDataset](#)

**Examples**

```
# need access to CDF (raw data) and ELU files
require(gcspikelite)
gcmsPath<-paste(find.package("gcspikelite"),"data",sep="/")

# full paths to file names
cdfFiles<-dir(gcmsPath,"CDF",full=TRUE)
eluFiles<-dir(gcmsPath,"ELU",full=TRUE)

# create a 'peaksDataset' object and add AMDIS peaks to it
pd<-peaksDataset(cdfFiles[1],mz=seq(50,550),rtrange=c(7.5,8.5))
pd<-addAMDISPeaks(pd,eluFiles[1])
```

---

addChromaTOFPeaks	<i>Add ChromaTOF peak detection results</i>
-------------------	---

---

**Description**

Reads ASCII tab-delimited format files (output from ChromaTOF) and attaches them to an already created peaksDataset object

**Usage**

```
addChromaTOFPeaks(
  object,
  fns = dir( "[Tt][Xx][Tx]" ),
  rtDivide = 60,
  verbose = TRUE,
  ...
)
```

**Arguments**

object	a peaksDataset object.
fns	character vector of same length as object@rawdata (user ensures the order matches)
rtDivide	number giving the amount to divide the retention times by.
verbose	whether to give verbose output, default TRUE
...	arguments passed on to parseChromaTOF

**Details**

Repeated calls to parseChromaTOF to add peak detection results to the original peaksDataset object.

**Value**

peaksDataset object

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[parseChromaTOF](#), [peaksDataset](#)

**Examples**

```
# need access to CDF (raw data) and ChromaTOF files
require(gcspikelite)
gcmsPath<-paste(find.package("gcspikelite"),"data",sep="/")

# full paths to file names
cdfFiles<-dir(gcmsPath,"CDF",full=TRUE)
# [not run] cTofFiles<-dir(gcmsPath,"txt",full=TRUE)

# create a 'peaksDataset' object and add ChromaTOF peaks to it
pd<-peaksDataset(cdfFiles[1],mz=seq(50,550),rtrange=c(7.5,8.5))
# [not run] pd<-addChromTOFPeaks(pd,...)
```

---

addXCMSPeaks

*addXCMSPeaks*

---

**Description**

Add xcms/CAMERA peak detection results

**Usage**

```
addXCMSPeaks(
  files,
  object,
  settings = list(),
  minintens = 100,
  minfeat = 6,
  BPPARAM = bpparam(),
  multipleMF = FALSE,
  multipleMFPParam = list(fwhm = c(5, 10, 15), mz.abs = 0.2, rt.abs = 2)
)
```

**Arguments**

files	list of chromatogram files
object	a peakDataset object
settings	see <a href="#">findPeaks-matchedFilter</a> <a href="#">findPeaks-centWave</a>
minintens	minimum ion intensity to be included into a pseudospectra
minfeat	minimum number of ion to be created a pseudospectra
BPPARAM	a parameter class specifying if and how parallel processing should be performed
multipleMF	logical Try to remove redundant peaks, in this case where there are any peaks within an absolute m/z value of 0.2 and within 3 s for any one sample in the xcmsSet (the largest peak is kept)
multipleMFParam	list. It contains the settings for the peak-picking. mz_abs represent the the mz range; rt_abs represent thert range
mz.abs	mz range
rt.abs	rt range

**Details**

Reads the raw data using xcms, group each extracted ion according to their retention time using CAMERA and attaches them to an already created peaksDataset object

Repeated calls to xcmsSet and annotate to perform peak-picking and deconvolution. The peak detection results are added to the original peaksDataset object. Two peak detection algorithms are available: continuous wavelet transform (peakPicking=c('cwt')) and the matched filter approach (peakPicking=c('mF')) described by Smith et al (2006). For further information consult the xcms package manual.

**Value**

peaksDataset object

**Author(s)**

Riccardo Romoli <[riccardo.romoli@unifi.it](mailto:riccardo.romoli@unifi.it)>

**See Also**

[peaksDataset](#) [findPeaks-matchedFilter](#) [findPeaks-centWave](#) [xcmsRaw-class](#)

**Examples**

```
files <- list.files(path = paste(find.package("gcspikelite"), "data",
                               sep = "/"), "CDF", full = TRUE)
data <- peaksDataset(files[1:2], mz = seq(50, 550), rtrange = c(7.5, 8.5))
## create settings object
mfp <- xcms::MatchedFilterParam(fwhm = 10, snthresh = 5)
cwt <- xcms::CentWaveParam()
data <- addXCMSPeaks(files[1:2], data, settings = mfp, multipleMF = FALSE)
data
```

---

betweenAlignment      *Data Structure for "between" alignment of many GCMS samples*

---

## Description

This function creates a "between" alignment (i.e. comparing merged peaks)

## Usage

```
betweenAlignment(
  pD,
  cAList,
  pAList,
  impList,
  filterMin = 1,
  gap = 0.7,
  D = 10,
  usePeaks = TRUE,
  df = 30,
  verbose = TRUE,
  metric = 2,
  type = 2,
  penalty = 0.2,
  compress = FALSE
)
```

## Arguments

pD	a peaksDataset object
cAList	list of clusterAlignment objects, one for each experimental group
pAList	list of progressiveAlignment objects, one for each experimental group
impList	list of imputation lists
filterMin	minimum number of peaks within a merged peak to be kept in the analysis
gap	gap parameter
D	retention time penalty parameter
usePeaks	logical, whether to use peaks (if TRUE) or the full 2D profile alignment (if FALSE)
df	distance from diagonal to calculate similarity
verbose	logical, whether to print information
metric	numeric, different algorithm to calculate the similarity matrix between two mass spectrum. metric=1 call normDotProduct(); metric=2 call ndpRT(); metric=3 call corPrt()
type	numeric, two different type of alignment function
penalty	penalization applied to the matching between two mass spectra if $(t_1 - t_2) > D$
compress	logical whether to compress the similarity matrix into a sparse format.

**Details**

betweenAlignment objects gives the data structure which stores the result of an alignment across several "pseudo" datasets. These pseudo datasets are constructed by merging the "within" alignments.

**Value**

betweenAlignment object

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[multipleAlignment](#)

**Examples**

```
require(gcspikelite)
## see 'multipleAlignment'
```

---

calcTimeDiffs

*Calculate retention time shifts from profile alignments*

---

**Description**

This function takes the set of all pairwise profile alignments and use these to estimate retention time shifts between each pair of samples. These will then be used to normalize the retention time penalty of the signal peak alignment.

**Usage**

```
calcTimeDiffs(pd, ca.full, verbose = TRUE)
```

**Arguments**

pd	a peaksDataset object
ca.full	a clusterAlignment object, fit with
verbose	logical, whether to print out information

**Details**

Using the set of profile alignments,

**Value**

list of same length as `ca.full@alignments` with the matrices giving the retention time penalties.

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[peaksAlignment](#), [clusterAlignment](#)

**Examples**

```
require(gcspikelite)

# paths and files
gcmsPath <- paste(find.package("gcspikelite"), "data", sep="/")
cdfFiles <- dir(gcmsPath, "CDF", full=TRUE)
eluFiles <- dir(gcmsPath, "ELU", full=TRUE)

# read data, peak detection results
pd <- peaksDataset(cdfFiles[1:2], mz=seq(50, 550), rtrange=c(7.5, 8.5))
pd <- addAMDISPeaks(pd, eluFiles[1:2])

# pairwise alignment using all scans
fullca <- clusterAlignment(pd, usePeaks=FALSE, df=100)

# calculate retention time shifts
timedf <- calcTimeDiffs(pd, fullca)
```

---

clusterAlignment	<i>Data Structure for a collection of all pairwise alignments of GCMS runs</i>
------------------	--

---

**Description**

Store the raw data and optionally, information regarding signal peaks for a number of GCMS runs

**Usage**

```
clusterAlignment(  
  pD,  
  runs = 1:length(pD@rawdata),  
  timedf = NULL,  
  usePeaks = TRUE,  
  verbose = TRUE,  
  ...  
)
```

**Arguments**

pD	a peaksDataset object.
runs	vector of integers giving the samples to calculate set of pairwise alignments over.
timedf	list (length = the number of pairwise alignments) of matrices giving the expected time differences expected at each pair of peaks used with usePeaks=TRUE, passed to peaksAlignment
usePeaks	logical, TRUE uses peakdata list, FALSE uses rawdata list for computing similarity.
verbose	logical, whether to print out info.
...	other arguments passed to peaksAlignment

**Details**

clusterAlignment computes the set of pairwise alignments.

**Value**

clusterAlignment object

**Author(s)**

Mark Robinson, Riccardo Romoli

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[peaksDataset](#), [peaksAlignment](#)

**Examples**

```
require(gcspikelite)

# paths and files
gcmsPath <- paste(find.package("gcspikelite"), "data", sep="/")
cdfFiles <- dir(gcmsPath, "CDF", full=TRUE)
eluFiles <- dir(gcmsPath, "ELU", full=TRUE)

# read data, peak detection results
pd <- peaksDataset(cdfFiles[1:2], mz=seq(50,550), rrange=c(7.5,8.5))
pd <- addAMDISPeaks(pd, eluFiles[1:2])

ca <- clusterAlignment(pd, gap=0.5, D=0.05, df=30, metric=1, type=1)
```

---

compress,peaksAlignment-method

*Compression method for peaksAlignment object*

---

**Description**

Compression method for peaksAlignment object

**Usage**

```
## S4 method for signature 'peaksAlignment'
compress(object, verbose = TRUE, ...)
```

**Arguments**

object	peaksAlignment
verbose	logical
...	further

**Author(s)**

MR

---

compress, progressiveAlignment-method  
*Compress method for progressiveAlignment*

---

**Description**

Decompress method for progressiveAlignment

**Usage**

```
## S4 method for signature 'progressiveAlignment'
compress(object, verbose = TRUE, ...)
```

**Arguments**

object	dummy
verbose	dummy
...	dummy

**Details**

Deompress method for progressiveAlignment

**Author(s)**

MR

---

corPrt *Retention Time Penalized Correlation*

---

**Description**

This function calculates the similarity of all pairs of peaks from 2 samples, using the spectra similarity and the retention time differences

**Usage**

```
corPrt(d1, d2, t1, t2, D, penalty = 0.2)
```

**Arguments**

d1	data matrix for sample 1
d2	data matrix for sample 2
t1	vector of retention times for sample 1
t2	vector of retention times for sample 2
D	retention time window for the matching
penalty	penalization applied to the matching between two mass spectra if $(t1-t2)>D$

**Details**

Computes the Pearson correlation between every pair of peak vectors in the retention time window (D) and returns the similarity matrix.

**Value**

matrix of similarities

**Author(s)**

Riccardo Romoli

**See Also**

[peaksAlignment](#)

**Examples**

```
## Not Run
require(gcspikelite)
files <- list.files(path = paste(find.package("gcspikelite"), "data",
                               sep = "/"), "CDF", full = TRUE)
data <- peaksDataset(files[1:2], mz = seq(50, 550), rtrange = c(7.5, 8.5))
## create settings object
mfp <- xcms::MatchedFilterParam(fwhm = 10, snthresh = 5)
cwt <- xcms::CentWaveParam()
data <- addXCMSPeaks(files[1:2], data, settings = mfp, multipleMF = FALSE)
data
## review peak picking
plotChrom(data, rtrange=c(7.5, 10.5), runs=c(1:2))

r <- corPrt(data@peaksdata[[1]], data@peaksdata[[2]],
            data@peaksrt[[1]], data@peaksrt[[2]], D = 50, penalty = 0.2)
## End (Not Run)
```

---

decompress,peaksAlignment-method

*Decompression method for peaksAlignment object*

---

**Description**

Decompression method for peaksAlignment object

**Usage**

```
## S4 method for signature 'peaksAlignment'
decompress(object, verbose = TRUE, ...)
```

**Arguments**

object	peaksAlignment object
verbose	dummy
...	dummy

**Author(s)**

MR

---

decompress,progressiveAlignment-method  
*Compress method for progressiveAlignment*

---

**Description**

Decompress method for progressiveAlignment

**Usage**

```
## S4 method for signature 'progressiveAlignment'  
decompress(object, verbose = TRUE, ...)
```

**Arguments**

object	progressiveAlignment object
verbose	logical
...	dummy

**Details**

Decompress method for progressiveAlignment

**Author(s)**

MR

---

deDuper	<i>deDuper</i>
---------	----------------

---

**Description**

Duplicate peak removal function

**Usage**

```
deDuper(object, mz.abs = 0.1, rt.abs = 2)
```

**Arguments**

object	xcms object
mz.abs	mz range
rt.abs	rt range

**Details**

Remove redundant peaks, in this case where there are any peaks within an absolute m/z value of 0.2 and within 3 s for any one sample in the xcmsSet (the largest peak is kept)

**Value**

an object of xcms class

**Author(s)**

r

---

distToLib	<i>distToLib</i>
-----------	------------------

---

**Description**

The function calculate the distance between each mas spec in the msp file and the aligned mass spec from each sampe

**Usage**

```
distToLib(mspLib, outList)
```

**Arguments**

mspLib	a .msp file from NIST
outList	an object from gatherInfo()

**Details**

Return the distance matrix

**Value**

the distance matrix between the mass spec and the aligned spec

**Author(s)**

Riccardo Romoli

---

dp

*Dynamic programming algorithm, given a similarity matrix*

---

**Description**

This function calls C code for a bare-bones dynamic programming algorithm, finding the best cost path through a similarity matrix.

**Usage**

```
dp(M, gap = 0.5, big = 1e+10, verbose = FALSE)
```

**Arguments**

M	similarity matrix
gap	penalty for gaps
big	large value used for matrix margins
verbose	logical, whether to print out information

**Details**

This is a pretty standard implementation of a bare-bones dynamic programming algorithm, with a single gap parameter and allowing only simple jumps through the matrix (up, right or diagonal).

**Value**

list with element match with the set of pairwise matches.

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**[normDotProduct](#)**Examples**

```
require(gcspikelite)

# paths and files
gcmsPath<-paste(find.package("gcspikelite"),"data",sep="/")
cdfFiles<-dir(gcmsPath,"CDF",full=TRUE)
eluFiles<-dir(gcmsPath,"ELU",full=TRUE)

# read data, peak detection results
pd<-peaksDataset(cdfFiles[1:2],mz=seq(50,550),rtrange=c(7.5,8.5))
pd<-addAMDISPeaks(pd,eluFiles[1:2])

# similarity matrix
r<-normDotProduct(pd@peaksdata[[1]],pd@peaksdata[[2]])

# dynamic-programming-based matching of peaks
v<-dp(r,gap=.5)
```

---

*dynRT**dynRT*

---

**Description**

Dynamic Retention Time Based Alignment algorithm, given a similarity matrix

**Usage**

```
dynRT(S)
```

**Arguments**

S                      similarity matrix

**Details**

This function align two chromatograms finding the maximum similarity among the mass spectra

**Value**

list containing the matched peaks between the two chromatograms. The number represent position of the spectra in the S matrix

**Author(s)**

riccardo.romoli@unifi.it

**Examples**

```

require(gcspikelite)
files <- list.files(path = paste(find.package("gcspikelite"), "data",
                               sep = "/"), "CDF", full = TRUE)
data <- peaksDataset(files[1:2], mz = seq(50, 550), rtrange = c(7.5, 8.5))
## create settings object
mfp <- xcms::MatchedFilterParam(fwhm = 10, snthresh = 5)
cwt <- xcms::CentWaveParam()
data <- addXCMSPeaks(files[1:2], data, settings = mfp, multipleMF = FALSE)
data
## review peak picking
plotChrom(data, rtrange=c(7.5, 10.5), runs=c(1:2))
## similarity
r <- ndpRT(data@peaksdata[[1]], data@peaksdata[[2]], data@peaksrt[[1]],
           data@peaksrt[[2]], D = 50)
## dynamic retention time based alignment algorithm
v <- dynRT(S = r)

```

---

eitherMatrix-class      *A class description*

---

**Description**

A class description

---

exportSpectra              *exportSpectra*

---

**Description**

Write the mass spectrum into a .msp file to be used in NIST search.

**Usage**

```
exportSpectra(object, outList, spectra, normalize = TRUE)
```

**Arguments**

object	an object of class "peaksDataset"
outList	an object created using the gatherInfo() function
spectra	numeric. The number of the mass spectra to be printed. It correspond to the number of the peak in the plot() and the number of the peak in the gatherInfo() list.
normalize	logical. If the mass spectra has to be normalized to 100

**Details**

Write the mass spectrum into a .msp file to be used in NIST search.

**Value**

a .msp file

**Author(s)**

riccardo.romoli@unifi.com

---

gatherInfo

*Gathers abundance informations from an alignment*


---

**Description**

Given an alignment table (indices of matched peaks across several samples) such as that within a progressiveAlignment or multipleAlignment object, this routines goes through the raw data and collects the abundance of each fragment peak, as well as the retention times across the samples.

**Usage**

```
gatherInfo(
  pD,
  obj,
  newind = NULL,
  method = c("apex"),
  findmzind = TRUE,
  useTIC = FALSE,
  top = NULL,
  intensity.cut = 0.05
)
```

**Arguments**

pD	a peaksDataset object, to get the abundance data from
obj	either a multipleAlignment or progressiveAlignment object
newind	list giving the
method	method used to gather abundance information, only apex implemented currently.
findmzind	logical, whether to take a subset of all m/z indices
useTIC	logical, whether to use total ion current for abundance summaries
top	only use the top top peaks
intensity.cut	percentage of the maximum intensity

**Details**

This procedure loops through the the table of matched peaks and gathers the

**Value**

Returns a list (of lists) for each row in the alignment table. Each list has 3 elements:

mz	a numerical vector of the m/z fragments used
rt	a numerical vector for the exact retention time of each peak across all samples
data	matrix of fragment intensities. If useTIC = TRUE, this matrix will have a single row

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[imputePeaks](#)

**Examples**

```
require(gcspikelite)

## paths and files
gcmsPath <- paste(find.package("gcspikelite"), "data", sep = "/")
cdfFiles <- dir(gcmsPath, "CDF", full = TRUE)
eluFiles <- dir(gcmsPath, "ELU", full = TRUE)

## read data, peak detection results
pd <- peaksDataset(cdfFiles[1:2], mz = seq(50, 550), rtrange = c(7.5, 8.5))
pd <- addAMDISPeaks(pd, eluFiles[1:2])

## multiple alignment
ma <- multipleAlignment(pd, c(1,1), wn.gap = 0.5, wn.D = 0.05, bw.gap = 0.6,
                        bw.D = 0.2, usePeaks = TRUE, filterMin = 1, df = 50,
                        verbose = TRUE, metric = 1, type = 1)

## gather apex intensities
d <- gatherInfo(pd, ma)

## table of retention times
nm <- list(paste("MP", 1:length(d), sep = ""), c("S1", "S2"))
rts <- matrix(unlist(sapply(d, .subset, "rt")), byrow = TRUE, nc = 2,
              dimnames = nm)
```

---

headToTailPlot	<i>Head to tail plot</i>
----------------	--------------------------

---

**Description**

The head-to-tail-plot for the mass spectra

**Usage**

```
headToTailPlot(specFromLib, specFromList)
```

**Arguments**

specFromLib	the mass spectra obtained from the .msp file
specFromList	the mass spectra obtained from <a href="#">gatherInfo</a>

**Details**

Head-to-tail-plot to visually compare the mass spectra

**Value**

the plot

**Author(s)**

Riccardo Romoli

---

importSpec	<i>importSpec</i>
------------	-------------------

---

**Description**

Read the mass spectra from an external msp file

**Usage**

```
importSpec(file)
```

**Arguments**

file	a .msp file from NIST search library database
------	---

**Details**

Read the mass spectra from an external file in msp format. The format is used in NIST search library database.

**Value**

list conaining the mass spectra

**Author(s)**

riccardo.romoli@unifi.it

---

imputePeaks

*Imputatin of locations of peaks that were undetected*

---

**Description**

Using the information within the peaks that are matched across several runs, we can impute the location of the peaks that are undetected in a subset of runs

**Usage**

```
imputePeaks(pD, obj, typ = 1, obj2 = NULL, filterMin = 1, verbose = TRUE)
```

**Arguments**

pD	a peaksDataset object
obj	the alignment object, either multipleAlignment or progressiveAlignment, that is used to infer the unmatched peak locations
typ	type of imputation to do, 1 for simple linear interpolation (default), 2 only works if obj2 is a clusterAlignment object
obj2	a clusterAlignment object
filterMin	minimum number of peaks within a merged peak to impute
verbose	logical, whether to print out information

**Details**

If you are aligning several samples and for a (small) subset of the samples in question, a peak is undetected, there is information within the alignment that can be useful in determining where the undetected peak is, based on the surrounding matched peaks. Instead of moving forward with missing values into the data matrices, this procedures goes back to the raw data and imputes the location of the apex (as well as the start and end), so that we do not need to bother with post-hoc imputation or removing data because of missing components.

We realize that imputation is prone to error and prone to attributing intensity from neighbouring peaks to the unmatched peak. We argue that this is still better than having to deal with these in statistical models after that fact. This may be an area of future improvement.

**Value**

list with 3 elements apex, start and end, each masked matrices giving the scan numbers of the imputed peaks.

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[multipleAlignment](#), [progressiveAlignment](#), [peaksDataset](#)

**Examples**

```
require(gcspikelite)

## paths and files
gcmsPath <- paste(find.package("gcspikelite"), "data", sep = "/")
cdfFiles <- dir(gcmsPath, "CDF", full = TRUE)
eluFiles <- dir(gcmsPath, "ELU", full = TRUE)

## read data, peak detection results
pd <- peaksDataset(cdfFiles[1:3], mz = seq(50,550), rtrange = c(7.5,8.5))
pd <- addAMDISPeaks(pd, eluFiles[1:3])

## alignments
ca <- clusterAlignment(pd, gap = 0.5, D = 0.05, df = 30, metric = 1, type =
  1, compress = FALSE)
pa <- progressiveAlignment(pd, ca, gap = 0.6, D = 0.1, df = 30,
  compress = FALSE)

v <- imputePeaks(pd, pa, filterMin = 1)
```

---

matchSpec

*matchSpec*

---

**Description**

Calculate the distance between a reference mass spectrum

**Usage**

```
matchSpec(spec1, outList, whichSpec)
```

**Arguments**

spec1	reference mass spectrum
outList	the return of <a href="#">gatherInfo</a>
whichSpec	the entry number of outList

**Details**

Calculate the distance between a reference mass spectrum and one from the sample

**Value**

the distance between the reference mass spectrum and the others

**Author(s)**

Riccardo Romoli

---

multipleAlignment-class

*Data Structure for multiple alignment of many GCMS samples*

---

**Description**

Store the raw data and optionally, information regarding signal peaks for a number of GCMS runs

**Usage**

```
multipleAlignment(  
  pd,  
  group,  
  bw.gap = 0.8,  
  wn.gap = 0.6,  
  bw.D = 0.2,  
  wn.D = 0.05,  
  filterMin = 1,  
  lite = FALSE,  
  usePeaks = TRUE,  
  df = 50,  
  verbose = TRUE,  
  timeAdjust = FALSE,  
  doImpute = FALSE,  
  metric = 2,  
  type = 2,  
  penalty = 0.2,  
  compress = FALSE  
)
```

**Arguments**

pd	a peaksDataset object
group	factor variable of experiment groups, used to guide the alignment algorithm
bw.gap	gap parameter for "between" alignments

wn.gap	gap parameter for "within" alignments
bw.D	distance penalty for "between" alignments. When type = 2 represent the retention time window expressed in seconds
wn.D	distance penalty for "within" alignments. When type = 2 represent the retention time window expressed in seconds
filterMin	minimum number of peaks within a merged peak to be kept in the analysis
lite	logical, whether to keep "between" alignment details (default, FALSE)
usePeaks	logical, whether to use peaks (if TRUE) or the full 2D profile alignment (if FALSE)
df	distance from diagonal to calculate similarity
verbose	logical, whether to print information
timeAdjust	logical, whether to use the full 2D profile data to estimate retention time drifts (Note: time required)
doImpute	logical, whether to impute the location of unmatched peaks
metric	numeric, different algorithm to calculate the similarity matrix between two mass spectrum. metric=1 call normDotProduct(); metric=2 call ndpRT(); metric=3 call corPrt()
type	numeric, two different type of alignment function
penalty	penalization applied to the matching between two mass spectra if $(t_1 - t_2) > D$
compress	logical whether to compress the similarity matrix into a sparse format.

### Details

multipleAlignment is the data structure giving the result of an alignment across several GCMS runs. Multiple alignments are done progressively. First, all samples with the same `tg$Group` label will be aligned (denoted a "within" alignment). Second, each group will be summarized into a pseudo-data set, essentially a spectrum and retention time for each matched peak of the within-alignment. Third, these "merged peaks" are aligned in the same progressive manner, here called a "between" alignment.

### Value

multipleAlignment object

### Author(s)

Mark Robinson

### References

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

### See Also

[peaksDataset](#), [betweenAlignment](#), [progressiveAlignment](#)

## Examples

```
require(gcspikelite)

## paths and files
gcmsPath <- paste(find.package("gcspikelite"), "data", sep = "/")
cdfFiles <- dir(gcmsPath, "CDF", full = TRUE)
eluFiles <- dir(gcmsPath, "ELU", full = TRUE)

## read data, peak detection results
pd <- peaksDataset(cdfFiles[1:2], mz = seq(50, 550), rtrange = c(7.5, 8.5))
pd <- addAMDISPeaks(pd, eluFiles[1:2])

## multiple alignment
ma <- multipleAlignment(pd, c(1, 1), wn.gap = 0.5, wn.D = 0.05, bw.gap = 0.6,
                        bw.D = 0.2, usePeaks = TRUE, filterMin = 1, df = 50,
                        verbose = TRUE, metric = 1, type = 1)
```

---

ndpRT

*Retention Time Penalized Normalized Dot Product*

---

## Description

This function calculates the similarity of all pairs of peaks from 2 samples, using the spectra similarity and the retention time differences

## Usage

```
ndpRT(s1, s2, t1, t2, D)
```

## Arguments

s1	data matrix for sample 1
s2	data matrix for sample 2
t1	vector of retention times for sample 1
t2	vector of retention times for sample 2
D	retention time window for the matching

## Details

Computes the normalized dot product between every pair of peak vectors in the retention time window (D) and returns a similarity matrix.

## Value

matrix of similarities

**Author(s)**

Riccardo Romoli

**See Also**[peaksAlignment](#)**Examples**

```
## Not Run
require(gcspikelite)
files <- list.files(path = paste(find.package("gcspikelite"), "data",
                               sep = "/"), "CDF", full = TRUE)
data <- peaksDataset(files[1:2], mz = seq(50, 550), rtrange = c(7.5, 8.5))
## create settings object
mfp <- xcms::MatchedFilterParam(fwhm = 10, snthresh = 5)
cwt <- xcms::CentWaveParam()
data <- addXCMSPeaks(files[1:2], data, settings = mfp, multipleMF = FALSE)
data
## review peak picking
plotChrom(data, rtrange = c(7.5, 10.5), runs = c(1:2))

r <- ndpRT(data@peaksdata[[1]], data@peaksdata[[2]],
           data@peaksrt[[1]], data@peaksrt[[2]], D = 50)
## End (Not Run)
```

---

normDotProduct

*Normalized Dot Product*

---

**Description**

This function calculates the similarity of all pairs of peaks from 2 samples, using the spectra similarity

**Usage**

```
normDotProduct(
  x1,
  x2,
  t1 = NULL,
  t2 = NULL,
  df = max(ncol(x1), ncol(x2)),
  D = 1e+05,
  timedf = NULL,
  verbose = FALSE
)
```

**Arguments**

x1	data matrix for sample 1
x2	data matrix for sample 2
t1	vector of retention times for sample 1
t2	vector of retention times for sample 2
df	distance from diagonal to calculate similarity
D	retention time penalty
timedf	matrix of time differences to normalize to. if NULL, 0 is used.
verbose	logical, whether to print out information

**Details**

Efficiently computes the normalized dot product between every pair of peak vectors and returns a similarity matrix. C code is called.

**Value**

matrix of similarities

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[dp](#), [peaksAlignment](#)

**Examples**

```
require(gcspikelite)

# paths and files
gcmsPath<-paste(find.package("gcspikelite"), "data", sep="/")
cdfFiles<-dir(gcmsPath, "CDF", full=TRUE)
eluFiles<-dir(gcmsPath, "ELU", full=TRUE)

# read data, peak detection results
pd<-peaksDataset(cdfFiles[1:2], mz=seq(50, 550), rtrange=c(7.5, 8.5))
pd<-addAMDISPeaks(pd, eluFiles[1:2])

r<-normDotProduct(pd@peaksdata[[1]], pd@peaksdata[[2]])
```

---

parseChromaTOF	<i>Parser for ChromaTOF files</i>
----------------	-----------------------------------

---

### Description

Reads ASCII ChromaTOF-format files from AMDIS (Automated Mass Spectral Deconvolution and Identification System)

### Usage

```
parseChromaTOF(  
  fn,  
  min.pc = 0.01,  
  mz = seq(85, 500),  
  rt.cut = 0.008,  
  rtrange = NULL,  
  skip = 1,  
  rtDivide = 60  
)
```

### Arguments

fn	ChromaTOF filename to read.
min.pc	minimum percent of maximum intensity.
mz	vector of mass-to-charge bins of raw data table.
rt.cut	the difference in retention time, below which peaks are merged together.
rtrange	retention time range to parse peaks from, can speed up parsing if only interested in a small region (must be numeric vector of length 2)
skip	number of rows to skip at beginning of the ChromaTOF
rtDivide	multiplier to divide the retention times by (default: 60)

### Details

parseChromaTOF will typically be called by [addChromaTOFPeaks](#), not called directly.

Peaks that are detected within `rt.cut` are merged together. This avoids peaks which are essentially overlapping.

Fragments that are less than `min.pc` of the maximum intensity fragment are discarded.

### Value

list with components `peaks` (table of spectra – rows are mass-to-charge and columns are the different detected peaks) and `tab` (table of features for each detection), according to what is stored in the ChromaTOF file.

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[addAMDISPeaks](#)

**Examples**

```
require(gcspikelite)

# paths and files
gcmsPath<-paste(find.package("gcspikelite"),"data",sep="/")
tofFiles<-dir(gcmsPath,"tof",full=TRUE)

# parse ChromaTOF file
cTofList<-parseChromaTOF(tofFiles[1])
```

---

parseELU

*Parser for ELU files*

---

**Description**

Reads ASCII ELU-format files from AMDIS (Automated Mass Spectral Deconvolution and Identification System)

**Usage**

```
parseELU(f, min.pc = 0.01, mz = seq(50, 550), rt.cut = 0.008, rtrange = NULL)
```

**Arguments**

f	ELU filename to read.
min.pc	minimum percent of maximum intensity.
mz	vector of mass-to-charge bins of raw data table.
rt.cut	the difference in retention time, below which peaks are merged together.
rtrange	retention time range to parse peaks from, can speed up parsing if only interested in a small region (must be numeric vector of length 2)

**Details**

parseELU will typically be called by [addAMDISPeaks](#), not called directly.

Peaks that are detected within `rt.cut` are merged together. This avoids peaks which are essentially overlapping.

Fragments that are less than `min.pc` of the maximum intensity fragment are discarded.

**Value**

list with components `peaks` (table of spectra – rows are mass-to-charge and columns are the different detected peaks) and `tab` (table of features for each detection), according to what is stored in the ELU file.

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[addAMDISPeaks](#)

**Examples**

```
require(gcspikelite)

# paths and files
gcmsPath<-paste(find.package("gcspikelite"),"data",sep="/")
eluFiles<-dir(gcmsPath,"ELU",full=TRUE)

# parse ELU file
eluList<-parseELU(eluFiles[1])
```

---

peaksAlignment-class *Data Structure for pairwise alignment of 2 GCMS samples*

---

**Description**

Store the raw data and optionally, information regarding signal peaks for a number of GCMS runs

**Usage**

```

peaksAlignment(
  d1,
  d2,
  t1,
  t2,
  gap = 0.5,
  D = 50,
  timedf = NULL,
  df = 30,
  verbose = TRUE,
  usePeaks = TRUE,
  compress = TRUE,
  metric = 2,
  type = 2,
  penalty = 0.2
)

```

**Arguments**

d1	matrix of MS intensities for 1st sample (if doing a peak alignment, this contains peak apexes/areas; if doing a profile alignment, this contains scan intensities. Rows are m/z bins, columns are peaks/scans.
d2	matrix of MS intensities for 2nd sample
t1	vector of retention times for 1st sample
t2	vector of retention times for 2nd sample
gap	gap penalty for dynamic programming algorithm. Not used if type=2
D	time window (on same scale as retention time differences, t1 and t2. Default scale is seconds.)
timedf	list (length = the number of pairwise alignments) of matrices giving the expected time differences expected at each pair of peaks used with usePeaks=TRUE.
df	integer, how far from the diagonal to go to calculate the similarity of peaks. Smaller value should run faster, but be careful not to choose too low.
verbose	logical, whether to print out info.
usePeaks	logical, TRUE uses peakdata list, FALSE uses rawdata list for computing similarity.
compress	logical, whether to compress the similarity matrix into a sparse format.
metric	numeric, different algorithm to calculate the similarity matrix between two mass spectrum. metric=1 call normDotProduct(); metric=2 call ndpRT(); metric=3 call corPrt()
type	numeric, two different type of alignment function
penalty	penalization applied to the matching between two mass spectra if (t1-t2)>D

**Details**

peaksAlignment is a hold-all data structure of the raw and peak detection data.

**Value**

peaksAlignment object

**Author(s)**

Mark Robinson, Riccardo Romoli

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[peaksDataset](#), [clusterAlignment](#)

**Examples**

```
## see clusterAlignment, it calls peaksAlignment

## Not Run:
files <- list.files(path = paste(find.package("gcspikelite"), "data",
                               sep = "/"), "CDF", full = TRUE)
data <- peaksDataset(files[1:2], mz = seq(50, 550), rtrange = c(7.5, 8.5))
## create settings object
mfp <- xcms::MatchedFilterParam(fwhm = 10, snthresh = 5)
cwt <- xcms::CentWaveParam(snthresh = 3, ppm = 3000, peakwidth = c(3, 40),
                           prefilter = c(3, 100), fitgauss = FALSE, integrate = 2, noise = 0,
                           extendLengthMSW = TRUE, mzCenterFun = "wMean")
data <- addXCMSPeaks(files[1:2], data, settings = mfp)
data
plotChrom(data, rtrange=c(7.5, 10.5), runs=c(1:2))

## align two chromatogram
pA <- peaksAlignment(data@peaksdata[[1]], data@peaksdata[[2]],
                    data@peaksrt[[1]], data@peaksrt[[2]], D = 50,
                    metric = 3, compress = FALSE, type = 2, penalty = 0.2)

plotAlignment(pA)
pA@v$match

par(mfrow=c(2,1))
plot(data@peaksdata[[1]][,15], type = 'h', main = paste(data@peaksrt[[1]][[15]]))
plot(data@peaksdata[[2]][,17], type = 'h',
     main = paste(data@peaksrt[[2]][[17]]))
## End (Not Run)
```

---

peaksDataset

*Data Structure for raw GCMS data and peak detection results*

---

## Description

Store the raw data and optionally, information regarding signal peaks for a number of GCMS runs

## Usage

```
peaksDataset(  
  fns = dir(, "[Cc][Dd][Ff]"),  
  verbose = TRUE,  
  mz = seq(50, 550),  
  rtDivide = 60,  
  rtrange = NULL  
)
```

## Arguments

fns	character vector, filenames of raw data in CDF format.
verbose	logical, if TRUE then iteration progress information is output.
mz	vector giving bins of raw data table.
rtDivide	number giving the amount to divide the retention times by.
rtrange	retention time range to limit data to (must be numeric vector of length 2)

## Details

peaksDataset is a hold-all data structure of the raw and peak detection data.

## Value

peaksDataset object

## Author(s)

Mark Robinson

## References

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**Examples**

```
require(gcspikelite)

# paths and files
gcmsPath<-paste(find.package("gcspikelite"), "data", sep="/")
cdfFiles<-dir(gcmsPath, "CDF", full=TRUE)
eluFiles<-dir(gcmsPath, "ELU", full=TRUE)

# read data
pd<-peaksDataset(cdfFiles[1:2], mz=seq(50, 550), rtrange=c(7.5, 8.5))
show(pd)
```

---

plotAlignedFrag      *plotAlignedFrag*

---

**Description**

Plot the aligned mass spectra

**Usage**

```
plotAlignedFrag(
  object,
  outList,
  specID,
  fullRange = TRUE,
  normalize = TRUE,
  ...
)
```

**Arguments**

object	where to keep the mass range of the experiment
outList	where to keep the mass spectra; both abundance than m/z
specID	a vector containing the index of the spectra to be plotted. Is referred to outList
fullRange	if TRUE uses the mass range of the whole experiment, otherwise uses only the mass range of each plotted spectrum
normalize	if TRUE normalize the intensity of the mass peak to 100, the most abundant is 100% and the other peaks are scaled consequentially
...	further arguments passed to the 'plot' command

**Details**

Plot the deconvoluted and aligned mass spectra collected using gatherInfo()

**Author(s)**

Riccardo Romoli (riccardo.romoli@unifi.it)

**Examples**

```
files <- list.files(path = paste(find.package("gcspikelite"), "data",
                               sep = "/"), "CDF", full = TRUE)
data <- peaksDataset(files[1:4], mz = seq(50, 550), rtrange = c(7.5, 8.5))
## create settings object
mfp <- xcms::MatchedFilterParam(fwhm = 10, snthresh = 5)
cwt <- xcms::CentWaveParam(snthresh = 3, ppm = 3000, peakwidth = c(3, 40),
  prefilter = c(3, 100), fitgauss = FALSE, integrate = 2, noise = 0,
  extendLengthMSW = TRUE, mzCenterFun = "wMean")
data <- addXCMSPeaks(files[1:4], data, settings = mfp)
data
## multiple alignment
ma <- multipleAlignment(data, c(1,1,2,2), wn.gap = 0.5, wn.D = 0.05,
  bw.gap = 0.6, bw.D = 0.2, usePeaks = TRUE, filterMin = 1, df = 50,
  verbose = TRUE, metric = 2, type = 2)

## gather apex intensities
gip <- gatherInfo(data, ma)
gip[[33]]
plotAlignedFragments(object = data, outList = gip, specID = 33)
```

---

*plotAlignment, peaksAlignment-method*  
*plotAlignment*

---

**Description**

Plotting functions for GCMS data objects

**Usage**

```
## S4 method for signature 'peaksAlignment'
plotAlignment(
  object,
  xlab = "Peaks - run 1",
  ylab = "Peaks - run 2",
  plotMatches = TRUE,
  matchPch = 19,
  matchLwd = 3,
  matchCex = 0.5,
  matchCol = "black",
  col = colorpanel(50, "white", "green", "navyblue"),
  breaks = seq(0, 1, length = 51),
  ...
)
```

## Arguments

object	a clusterAlignment object
xlab	x-axis label
ylab	y-axis label
plotMatches	logical, whether to plot matches
matchPch	match plotting character
matchLwd	match line width
matchCex	match character expansion factor
matchCol	match colour
col	vector of colours for colour scale
breaks	vector of breaks for colour scale
...	further arguments passed to image

## Details

Plot an object of [peaksAlignment](#)

The similarity matrix is plotted and optionally, the set of matching peaks. clusterAlignment objects are just a collection of all pairwise peakAlignment objects.

## Value

plot an object of class [peaksAlignment](#)

## Author(s)

Mark Robinson

## References

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

## See Also

[peaksAlignment](#) [plotAlignment](#)

## Examples

```
require(gcspikelite)
files <- list.files(path = paste(find.package("gcspikelite"), "data",
                               sep = "/"), "CDF", full = TRUE)
data <- peaksDataset(files[1:2], mz = seq(50, 550), rtrange = c(7.5, 8.5))
## create settings object
mfp <- xcms::MatchedFilterParam(fwhm = 10, snthresh = 5)
cwt <- xcms::CentWaveParam(snthresh = 3, ppm = 3000, peakwidth = c(3, 40),
                           prefilter = c(3, 100), fitgauss = FALSE, integrate = 2, noise = 0,
                           extendLengthMSW = TRUE, mzCenterFun = "wMean")
```

```

data <- addXCMSPeaks(files[1:2], data, settings = mfp)
data
## image plot
plotChrom(data, rtrange = c(7.5,8.5), plotPeaks = TRUE, plotPeakLabels =TRUE)

## align two chromatogram
pA <- peaksAlignment(data@peaksdata[[1]], data@peaksdata[[2]],
                     data@peaksrt[[1]], data@peaksrt[[2]], D = 50,
                     compress = FALSE, type = 1, metric = 1,
                     gap = 0.5)
plotAlignment(pA)

```

---

plotChrom,peaksDataset-method

*Plotting functions for GCMS data objects*

---

## Description

Store the raw data and optionally, information regarding signal peaks for a number of GCMS runs

## Usage

```

## S4 method for signature 'peaksDataset'
plotChrom(
  object,
  runs = 1:length(object@rawdata),
  mzind = 1:nrow(object@rawdata[[1]]),
  mind = NULL,
  plotSampleLabels = TRUE,
  calcGlobalMax = FALSE,
  peakCex = 0.8,
  plotPeaks = TRUE,
  plotPeakBoundaries = FALSE,
  plotPeakLabels = FALSE,
  plotMergedPeakLabels = TRUE,
  mlwd = 3,
  usePeaks = TRUE,
  plotAcrossRuns = FALSE,
  overlap = F,
  rtrange = NULL,
  cols = NULL,
  thin = 1,
  max.near = median(object@rawrt[[1]]),
  how.near = 50,
  scale.up = 1,
  ...
)

```

**Arguments**

object	a peaksDataset object.
runs	set of run indices to plot
mzind	set of mass-to-charge indices to sum over (default, all)
mind	matrix of aligned indices
plotSampleLabels	logical, whether to display sample labels
calcGlobalMax	logical, whether to calculate an overall maximum for scaling
peakCex	character expansion factor for peak labels
plotPeaks	logical, whether to plot hashes for each peak
plotPeakBoundaries	logical, whether to display peak boundaries
plotPeakLabels	logical, whether to display peak labels
plotMergedPeakLabels	logical, whether to display 'merged' peak labels
mlwd	line width of lines indicating the alignment
usePeaks	logical, whether to plot alignment of peaks (otherwise, scans)
plotAcrossRuns	logical, whether to plot across peaks when unmatched peak is given
overlap	logical, whether to plot TIC/XICs overlapping
rtrange	vector of length 2 giving start and end of the X-axis
cols	vector of colours (same length as the length of runs)
thin	when usePeaks=FALSE, plot the alignment lines every thin values
max.near	where to look for maximum
how.near	how far away from max.near to look
scale.up	a constant factor to scale the TICs
...	further arguments passed to the plot

**Details**

Each TIC is scale to the maximum value (as specified by the how.near and max.near values). The many parameters gives considerable flexibility of how the TICs can be visualized.

**Value**

plot the chromatograms

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**[peaksDataset](#)**Examples**

```
require(gcspikelite)

## paths and files
gcmsPath <- paste(find.package("gcspikelite"), "data", sep="/")
cdfFiles <- dir(gcmsPath, "CDF", full=TRUE)
eluFiles <- dir(gcmsPath, "ELU", full=TRUE)

## read data
pd <- peaksDataset(cdfFiles[1:3], mz=seq(50,550), rrange=c(7.5,8.5))

## image plot
plotChrom(pd, rrange = c(7.5,8.5), plotPeaks = TRUE,
          plotPeakLabels = TRUE)
```

---

*plotClustAlignment, clusterAlignment-method*  
*plotClustAlignment*

---

**Description**

Plotting functions for GCMS data objects

**Usage**

```
## S4 method for signature 'clusterAlignment'
plotClustAlignment(object, alignment = 1, ...)
```

**Arguments**

<code>object</code>	clusterAlignment object.
<code>alignment</code>	the set of alignments to plot
<code>...</code>	further arguments passed to image. See also plotAlignment

**Details**

For clusterAlignment objects, the similarity matrix is plotted and optionally, the set of matching peaks. clusterAlignment objects are just a collection of all pairwise peakAlignment objects.

**Value**

plot the pairwise alignment

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[plotAlignment](#)

**Examples**

```
require(gcspikelite)

# paths and files
gcmsPath <- paste(find.package("gcspikelite"), "data", sep="/")
cdfFiles <- dir(gcmsPath, "CDF", full=TRUE)
eluFiles <- dir(gcmsPath, "ELU", full=TRUE)

# read data, peak detection results
pd <- peaksDataset(cdfFiles[1:2], mz=seq(50,550), rtrange=c(7.5,8.5))
pd <- addAMDISPeaks(pd, eluFiles[1:2])

ca <- clusterAlignment(pd, gap=0.5, D=0.05, df=30, metric=1, type=1)
plotClustAlignment(ca, run = 1)
plotClustAlignment(ca, run = 2)
plotClustAlignment(ca, run = 3)
```

---

plotFrag

*plotFrag*

---

**Description**

Plot the mass spectra from the profile matrix

**Usage**

```
plotFrag(object, sample, specID, normalize = TRUE, ...)
```

**Arguments**

object	an object of class "peaksDataset" where to keep the mass spectra; both abundance (y) than m/z (x)
sample	character, the sample from were to plot the mass spectra
specID	numerical, a vector containing the index of the spectra to be plotted.

normalize        logical, if TRUE normalize the intensity of the mass peak to 100, the most abundant is 100 consequentially

...                other parameter passed to the plot() function

### Details

Plot the deconvoluted mass spectra from the profile matrix

### Author(s)

riccardo.romoli@unifi.it

### Examples

```
files <- list.files(path = paste(find.package("gcspikelite"), "data",
                               sep = "/"), "CDF", full = TRUE)
data <- peaksDataset(files[1:2], mz = seq(50, 550), rtrange = c(7.5, 8.5))
## create settings object
mfp <- xcms::MatchedFilterParam(fwhm = 10, snthresh = 5)
cwt <- xcms::CentWaveParam(snthresh = 3, ppm = 3000, peakwidth = c(3, 40),
                           prefilter = c(3, 100), fitgauss = FALSE, integrate = 2, noise = 0,
                           extendLengthMSW = TRUE, mzCenterFun = "wMean")
data <- addXCMSPeaks(files[1:2], data, settings = mfp)
data
## align two chromatogram
pA <- peaksAlignment(data@peaksdata[[1]], data@peaksdata[[2]],
                    data@peaksrt[[1]], data@peaksrt[[2]], D = 50,
                    metric = 3, compress = FALSE, type = 2, penalty = 0.2)
pA@v$match
## plot the mass spectra
par(mfrow=c(2,1))
plotFrag(object=data, sample=1, specID=10)
plotFrag(object=data, sample=2, specID=12)
```

---

plotImage

*Plot of images of GCMS data*

---

### Description

Image plots (i.e. 2D heatmaps) of raw GCMS profile data

### Usage

```
## S4 method for signature 'peaksDataset'
plotImage(
  object,
  run = 1,
  rtrange = c(11, 13),
```

```
main = NULL,  
mzrange = c(50, 200),  
SCALE = log2,  
...  
)
```

### Arguments

object	a peaksDataset object
run	index of the run to plot an image for
rtrange	vector of length 2 giving start and end of the X-axis (retention time)
main	main title (auto-constructed if not specified)
mzrange	vector of length 2 giving start and end of the Y-axis (mass-to-charge ratio)
SCALE	function called to scale the data (default: log2)
...	further arguments passed to the image command

### Details

For peakDataset objects, each TIC is scale to the maximum value (as specified by the `how.near` and `max.near` values). The many parameters gives considerable flexibility of how the TICs can be visualized.

For peakAlignment objects, the similarity matrix is plotted and optionally, the set of matching peaks. `clusterAlignment` objects are just a collection of all pairwise peakAlignment objects.

### Author(s)

Mark Robinson

### References

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

### See Also

[plot](#), [peaksDataset](#)

### Examples

```
require(gcspikelite)  
  
# paths and files  
gcmsPath<-paste(find.package("gcspikelite"),"data",sep="/")  
cdfFiles<-dir(gcmsPath,"CDF",full=TRUE)  
eluFiles<-dir(gcmsPath,"ELU",full=TRUE)  
  
# read data  
pd<-peaksDataset(cdfFiles[1],mz=seq(50,550),rtrange=c(7.5,8.5))
```

```
# image plot
plotImage(pd,run=1,rtrange=c(7.5,8.5),main="")
```

---

progressiveAlignment-class

*Data Structure for progressive alignment of many GCMS samples*

---

## Description

Performs a progressive peak alignment (clustalw style) of multiple GCMS peak lists

## Usage

```
progressiveAlignment(
  pD,
  cA,
  D = 50,
  gap = 0.5,
  verbose = TRUE,
  usePeaks = TRUE,
  df = 30,
  compress = FALSE,
  type = 2
)
```

## Arguments

pD	a peaksDataset object
cA	a clusterAlignment object
D	retention time penalty
gap	gap parameter
verbose	logical, whether to print information
usePeaks	logical, whether to use peaks (if TRUE) or the full 2D profile alignment (if FALSE)
df	distance from diagonal to calculate similarity
compress	logical, whether to store the similarity matrices in sparse form
type	numeric, two different type of alignment function

## Details

The progressive peak alignment we implemented here for multiple GCMS peak lists is analogous to how `clustalw` takes a set of pairwise sequence alignments and progressively builds a multiple alignment. More details can be found in the reference below.

**Value**

progressiveAlignment object

**Author(s)**

Mark Robinson

**References**

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

**See Also**

[peaksDataset](#), [multipleAlignment](#)

**Examples**

```
require(gcspikelite)
files <- list.files(path = paste(find.package("gcspikelite"), "data",
                               sep = "/"), "CDF", full = TRUE)
data <- peaksDataset(files[1:2], mz = seq(50, 550), rtrange = c(7.5, 8.5))
## create settings object
mfp <- xcms::MatchedFilterParam(fwhm = 10, snthresh = 5)
cwt <- xcms::CentWaveParam(snthresh = 3, ppm = 3000, peakwidth = c(3, 40),
                           prefilter = c(3, 100), fitgauss = FALSE, integrate = 2, noise = 0,
                           extendLengthMSW = TRUE, mzCenterFun = "wMean")
data <- addXCMSPeaks(files[1:2], data, settings = mfp)
data
ca <- clusterAlignment(data, gap = 0.5, D = 0.05, df = 30, metric = 1,
                       type = 1, compress = FALSE)
pa <- progressiveAlignment(data, ca, gap = 0.6, D = 0.1, df = 30,
                           type = 1, compress = FALSE)
```

---

retFatMatrix

*retFatMatrix*

---

**Description**

Build a fat data matrix

**Usage**

```
retFatMatrix(object, data, minFilter = round(length(object@files)/3 * 2))
```

## Arguments

object	peakDataset object
data	a gatherInfo() object
minFilter	the minimum number for a feature to be returned in the data matrix. Default is 2/3 of the samples

## Details

This function allows to extract the data from an object created using `gatherInfo` and build a data matrix using the area of the deconvoluted and aligned peaks. The row are the samples while the column represent the different peaks.

## Value

A fat data matrix containing the area of the deconvoluted and aligned peaks. The row are the samples while the column represent the different peaks

## Author(s)

Riccardo Romoli <[riccardo.romoli@unifi.it](mailto:riccardo.romoli@unifi.it)>

## See Also

[gatherInfo](#)

## Examples

```
require(gcspikelite)
files <- list.files(path = paste(find.package("gcspikelite"), "data",
                               sep = "/"), "CDF", full = TRUE)
data <- peaksDataset(files[1:2], mz = seq(50, 550), rtrange = c(7.5, 8.5))
## create settings object
mfp <- xcms::MatchedFilterParam(fwhm = 10, snthresh = 5)
cwt <- xcms::CentWaveParam(snthresh = 3, ppm = 3000, peakwidth = c(3, 40),
                           prefilter = c(3, 100), fitgauss = FALSE, integrate = 2, noise = 0,
                           extendLengthMSW = TRUE, mzCenterFun = "wMean")
data <- addXCMSPeaks(files[1:2], data, settings = mfp)
data
ma <- multipleAlignment(pd = data, group = c(1,1),
                       filterMin = 1, metric = 2, type = 2)
outList <- gatherInfo(data, ma)
mtxD <- retFatMatrix(object = data, data = outList, minFilter = 1)
```

---

`rmaFitUnit`*Fits a robust linear model (RLM) for one metabolite*

---

**Description**

Using `r1m` from MASS, this procedure fits a linear model using all the fragments

**Usage**

```
rmaFitUnit(  
  u,  
  maxit = 5,  
  mzEffect = TRUE,  
  cls = NULL,  
  fitSample = TRUE,  
  fitOrCoef = c("coef", "fit"),  
  TRANSFORM = log2  
)
```

**Arguments**

<code>u</code>	a metabolite unit (list object with vectors <code>mz</code> and <code>rt</code> for <code>m/z</code> and retention times, respectively and a data element giving the <code>fragmentxsample</code> intensity matrix)
<code>maxit</code>	maximum number of iterations (default: 5)
<code>mzEffect</code>	logical, whether to fit <code>m/z</code> effect (default: TRUE)
<code>cls</code>	class variable
<code>fitSample</code>	whether to fit individual samples (alternative is fit by group)
<code>fitOrCoef</code>	whether to return a vector of coefficients (default: "coef"), or an <code>r1m</code> object ("fit")
<code>TRANSFORM</code>	function to transform the raw data to before fitting (default: <code>log2</code> )

**Details**

Fits a robust linear model.

**Value**

list giving elements of fragment and sample coefficients (if `fitOrCoef="coef"`) or a list of elements from the fitting process (if `fitOrCoef="fit"`)

**Author(s)**

Mark Robinson

## References

Mark D Robinson (2008). Methods for the analysis of gas chromatography - mass spectrometry data *PhD dissertation* University of Melbourne.

## See Also

[peaksAlignment](#), [clusterAlignment](#)

## Examples

```
require(gcspikelite)

# paths and files
gcmsPath<-paste(find.package("gcspikelite"),"data",sep="/")
cdfFiles<-dir(gcmsPath,"CDF",full=TRUE)
eluFiles<-dir(gcmsPath,"ELU",full=TRUE)

# read data, peak detection results
pd<-peaksDataset(cdfFiles[1:2],mz=seq(50,550),rtrange=c(7.5,8.5))
pd<-addAMDISPeaks(pd,eluFiles[1:2])

# pairwise alignment using all scans
fullca<-clusterAlignment(pd, usePeaks = FALSE, df = 100)

# calculate retention time shifts
timedf<-calcTimeDiffs(pd, fullca)
```

---

show,multipleAlignment-method

*Store the raw data and optionally, information regarding signal peaks  
for a number of GCMS runs*

---

## Description

multipleAlignment is the data structure giving the result of an alignment across several GCMS runs. Multiple alignments are done progressively. First, all samples with the same `tg$Group` label will be aligned (denoted a "within" alignment). Second, each group will be summarized into a pseudo-data set, essentially a spectrum and retention time for each matched peak of the within-alignment. Third, these "merged peaks" are aligned in the same progressive manner, here called a "between" alignment.

## Usage

```
## S4 method for signature 'multipleAlignment'  
show(object)
```

**Arguments**

object            multipleAlignment object

**Author(s)**

Mark Robinson

# Index

- \* **classes**
  - betweenAlignment, 7
  - clusterAlignment, 9
  - multipleAlignment-class, 24
  - peaksAlignment-class, 31
  - peaksDataset, 34
  - plotAlignment, peaksAlignment-method, 36
  - plotChrom, peaksDataset-method, 38
  - plotClustAlignment, clusterAlignment-method, 40
  - plotImage, 42
  - progressiveAlignment-class, 44
- \* **gatherInfo()**
  - plotAlignedFragments, 35
- \* **internal**
  - compress, peaksAlignment-method, 11
  - compress, progressiveAlignment-method, 12
  - decompress, peaksAlignment-method, 13
  - decompress, progressiveAlignment-method, 14
- \* **manip**
  - addAMDISPeaks, 3
  - addChromaTOFPeaks, 4
  - addXCMSPeaks, 5
  - calcTimeDiffs, 8
  - corPrt, 12
  - dp, 16
  - gatherInfo, 19
  - imputePeaks, 22
  - ndpRT, 26
  - normDotProduct, 27
  - parseChromaTOF, 29
  - parseELU, 30
  - rmaFitUnit, 47
- \* **plot()**
  - plotAlignedFragments, 35
  - addAMDISPeaks, 3, 30, 31
  - addChromaTOFPeaks, 4, 29
  - addXCMSPeaks, 5
  - betweenAlignment, 7, 25
  - betweenAlignment-class
    - (betweenAlignment), 7
  - betweenAlignment-method
    - (betweenAlignment), 7
  - betweenAlignment-show
    - (betweenAlignment), 7
  - calcTimeDiffs, 8
  - clusterAlignment, 9, 9, 33, 48
  - clusterAlignment-class
    - (clusterAlignment), 9
  - clusterAlignment-plot
    - (clusterAlignment), 9
  - clusterAlignment-show
    - (clusterAlignment), 9
  - compress, peaksAlignment-method, 11
  - compress, progressiveAlignment-method, 12
  - corPrt, 12
  - decompress, peaksAlignment-method, 13
  - decompress, progressiveAlignment-method, 14
  - deDuper, 15
  - distToLib, 15
  - dp, 16, 28
  - dynRT, 17
  - eitherMatrix-class, 18
  - exportSpectra, 18
  - gatherInfo, 19, 21, 23, 46
  - headToTailPlot, 21
  - importSpec, 21

- imputePeaks, [20](#), [22](#)
- matchSpec, [23](#)
- multipleAlignment, [8](#), [23](#), [45](#)
- multipleAlignment
  - (multipleAlignment-class), [24](#)
- multipleAlignment-class, [24](#)
- multipleAlignment-class,
  - (multipleAlignment-class), [24](#)
- multipleAlignment-method
  - (multipleAlignment-class), [24](#)
- multipleAlignment-show,
  - (multipleAlignment-class), [24](#)
  
- ndpRT, [26](#)
- normDotProduct, [17](#), [27](#)
  
- parseChromaTOF, [5](#), [29](#)
- parseELU, [3](#), [30](#)
- peaksAlignment, [9](#), [10](#), [13](#), [27](#), [28](#), [37](#), [48](#)
- peaksAlignment (peaksAlignment-class),
  - [31](#)
- peaksAlignment-class, [31](#)
- peaksAlignment-plot
  - (peaksAlignment-class), [31](#)
- peaksAlignment-show
  - (peaksAlignment-class), [31](#)
- peaksDataset, [3](#), [5](#), [6](#), [10](#), [23](#), [25](#), [33](#), [34](#), [40](#),
  - [43](#), [45](#)
- peaksDataset-class (peaksDataset), [34](#)
- peaksDataset-plot (peaksDataset), [34](#)
- peaksDataset-show (peaksDataset), [34](#)
- plot, [43](#)
- plot, clusterAlignment, ANY-method
  - (clusterAlignment), [9](#)
- plot, clusterAlignment-method
  - (clusterAlignment), [9](#)
- plot, peaksAlignment, ANY-method
  - (peaksAlignment-class), [31](#)
- plot, peaksAlignment-method
  - (peaksAlignment-class), [31](#)
- plot, peaksDataset, ANY-method
  - (peaksDataset), [34](#)
- plot, peaksDataset-method
  - (peaksDataset), [34](#)
- plotAlignedFragments, [35](#)
- plotAlignment, [37](#), [41](#)
- plotAlignment, peaksAlignment-method,
  - [36](#)
- plotChrom, peaksDataset-method, [38](#)
- plotClustAlignment, clusterAlignment-method,
  - [40](#)
- plotFragments, [41](#)
- plotImage, [42](#)
- plotImage, peaksDataset-method
  - (plotImage), [42](#)
- progressiveAlignment, [23](#), [25](#)
- progressiveAlignment
  - (progressiveAlignment-class),
    - [44](#)
- progressiveAlignment-class, [44](#)
- progressiveAlignment-show
  - (progressiveAlignment-class),
    - [44](#)
  
- retFatMatrix, [45](#)
- rmaFitUnit, [47](#)
  
- show, (betweenAlignment), [7](#)
- show, clusterAlignment-method
  - (clusterAlignment), [9](#)
- show, multipleAlignment-method, [48](#)
- show, peaksAlignment-method
  - (peaksAlignment-class), [31](#)
- show, peaksDataset-method
  - (peaksDataset), [34](#)
- show, progressiveAlignment-method
  - (progressiveAlignment-class),
    - [44](#)