Package ‘flagme’

March 28, 2017

Version 1.30.0
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 - mass spectrometry metabolomics data
License LGPL (>= 2)
Collate 0classes.R clusterAlignment.R init.R multipleAlignment.R
    rmaFitUnit.R addXCMSPeaks.R correlationAlignment.R
biocViews DifferentialExpression, MassSpectrometry
NeedsCompilation yes

R topics documented:

addAMDISPeaks .................................................. 2
addChromaTOFPeaks ........................................... 3
addXCMSPeaks .................................................. 4
betweenAlignment ............................................. 5
calcTimeDiffs .................................................. 6
clusterAlignment .............................................. 7
compress ......................................................... 9
correlationAlignment ....................................... 10
correlationAlignment-class ................................ 11
dp ................................................................. 12
eitherMatrix-class .......................................... 13
exportSpectra ................................................ 13
gatherInfo .................................................... 14
imputePeaks ................................................... 15
Description

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

Usage

```r
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

See Also
def sexually

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),rt=range=c(7.5,8.5))
peaksDataset(pd,eluFiles[1])

addChromaTOFPeaks

Add ChromaTOF peak detection results

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][Tt]"),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
addXCMSPeaks

**Description**

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.

**Usage**

```r
addXCMSPeaks(files, object, peakPicking=c('cwt', 'mF'), ...)
```

**Arguments**

- **files**: character vector of same length as object@rawdata (user ensures the order matches)
- **object**: a peaksDataset object.
- **peakPicking**: Methods to use for peak detection. See details.
- **...**: arguments passed on to xcmsSet and annotate

**Details**

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.

**Examples**

```r
# 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,...)
```
betweenAlignment

**Value**

peaksDataset object

**Author(s)**

Riccardo Romoli <riccardo.romoli@unifi.it>

**See Also**

peaksDataset findPeaks.matchedFilter findPeaks.centWave xcmsRaw-class

**Examples**

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

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

# create a 'peaksDataset' object and add XCMS peaks to it
pd <- peaksDataset(cdfFiles[1], mz=seq(50,550), rtrange=c(7.5,8.5))
pd <- addXCMSPeaks(cdfFiles[1], pd, peakPicking=c("mF"),
   snthresh=3, fwhm=4, step=1, steps=2, mzdif=0.5)
```

---

**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=3,gap=0.7,D=10,usePeaks=TRUE,df=30,verbose=TRUE)
```

**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
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


See Also

multipleAlignment

Examples

```r
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

```r
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,
clusterAlignment

Value

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

Author(s)

Mark Robinson

References


See Also

`peaksAlignment`, `clusterAlignment`

Examples

```r
require(gcspikelite)

# paths and files
gcmsPath <- paste(find.package("gcspikelite"),"data",sep="/")
cdffFiles <- 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)
```

class(clusterAlignment)  

Data Structure for a collection of all pairwise alignments of GCMS runs

Description

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

Usage

```r
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

references


see also

peaksDataset, peaksAlignment

examples

require(gcspikelite)

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

# read data, peak detection results
pd<-peaksDataset(cddfFiles[1:2],mz=seq(50,550),rtrange=c(7.5,8.5))
pd<-addAMDISPeaks(pd,eluFiles[1:2])
ca<-clusterAlignment(pd, gap = .5,D=.05,df=30)
Compress an alignment object

**Description**

Many of the peaks are not similar. So, the set of pairwise similarity matrices can be compressed.

**Usage**

```r
compress(object, verbose=TRUE, ...) 
```

`decompress(object, verbose=TRUE, ...)`

**Arguments**

- `object`  
  - a `peaksAlignment`, `peaksAlignment` or `peaksAlignment` object to be compressed
- `verbose`  
  - logical, whether to print out information
- `...`  
  - further arguments

**Details**

Using sparse matrix representations, a significant compression can be achieved. Here, we use the `matrix.csc` class of the `SpareM` package.

**Value**

- an object of the same type as the input object

**Author(s)**

Mark Robinson

**References**


**See Also**

- `peaksAlignment`, `clusterAlignment`, `progressiveAlignment`

**Examples**

```r
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))
```
correlationAlignment

Function for correlation-based alignment strategy of multiple GCMS samples

Description
Performs a correlation-based peak alignment of multiple GCMS peak lists; this function is able to align multiple samples, by a center-star strategy.

Usage
correlationAlignment(object, thr=0.85, D=20, penalty=0.2, normalize=TRUE, minFilter=1)

Arguments
- **object**: a peaksDataset object
- **thr**: correlation threshold from 0 (min) to 1 (max)
- **D**: retention time window in seconds
- **penalty**: the penalty inflicted to a match between two peaks when the retention time difference exceed the parameter D
- **normalize**: logical, whether to use normalized-to-100 peaks intensity or as such
- **minFilter**: if a feature is matched in a number of samples less than minFilter, this feature is trashed. The value of minFilter must be smaller than the number of samples

Details
The correlation-based peak alignment for multiple GCMS peak lists uses a center-star technique to the alignment of the peaks. The combination of the D and penalty parameters allow the users to force the algorithm to match the peaks close to the reference. The thr parameter control the matching factor.

Value
correlationAlignment object

Author(s)
Riccardo Romoli <riccardo.romoli@unifi.it>

See Also
peaksDataset, addXCMSPeaks, correlationAlignment-class
Examples

```r
require(gcspikelite)

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

# read data, peak detection results
pd <- peaksDataset(cdfFiles[1:2], mz=seq(50,550), rtrange=c(7.5,8.5))
pd <- addXCMSpeaks(files=cdfFiles[1:2], object=pd, peakPicking=c("mF"),
                   snthresh=3, fwhm=4, step=1, steps=2, mzdiff=0.5)
mp <- correlationAlignment(object=pd, thr=0.85, D=20, penalty=0.2,
                           normalize=TRUE, minFilter=1)
```

Description

A class containing the index of the aligned chromatographic peaks.

Objects from the Class

Objects can be created by calls of the form `new("correlationAlignment", ...). The object created contains both the results of the alignment procedure and the file used and as center-star.

Slots

Alignment: Object of class "align" contain the matrix of the aligned features. The rows represent the different peaks while the columns represent the files. The values of the matrix refers to `peaksind` slot of the `peaksDataSet` object.

Center: Object of class "character" contain the file used as a center-star.

Methods

show signature(object = "correlationAlignment")

Author(s)

Riccardo Romoli <riccardo.romoli@unifi.it>

See Also

`correlationAlignment` for further information about the alignment function.

Examples

showClass("correlationAlignment")
**Description**

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

**Usage**

```r
dp(M, gap=.5, big=10000000000, 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**


**See Also**

`normDotProduct`

**Examples**

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

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

---

### eitherMatrix-class

**The `eitherMatrix` class**

**Description**

A container to store either matrix or matrix.csc objects

**Author(s)**

Mark Robinson

**References**


**See Also**

`peaksAlignment`

---

### exportSpectra

**exportSpectra**

**Description**

Write the deconvoluted mass spectra to an external file

**Usage**

```r
exportSpectra(object, sample, spectraID, 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
- **spectraID**: 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 other peaks are scaled consequetially
Details

Write a .msp file of the deconvoluted mass spectra. Useful to try to identify the unknown spectra using NIST Search.

Value

a .msp file ready to be read using NIST search

Author(s)

riccardo.romoli@unifi.it

---

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 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
imputePeaks

Author(s)

Mark Robinson

References


See Also

imputePeaks

Examples

```r
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=.05, bw.gap=0.6, bw.D=.2, usePeaks=TRUE, filterMin=1, df=50, verbose=TRUE)

d <- gatherInfo(pd, ma)

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

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, type = 1, obj2 = NULL, filterMin = 3, verbose = TRUE)`

Arguments

- `pD`: a `peaksDataset` object
- `obj`: the alignment object, either `multipleAlignment` or `progressiveAlignment`, that is used to infer the unmatched peak locations
imputePeaks

type  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


See Also

multipleAlignment, progressiveAlignment, peaksDataset

Examples

```r
require(gcspikelite)

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

cdffFiles <- dir(gcmsPath,"CDF",full=TRUE)
eluFiles <- dir(gcmsPath,"ELU",full=TRUE)

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

# alignments
ca <- clusterAlignment(pd, gap = .5,D=.05,df=30)
pa <- progressiveAlignment(pd, ca, gap = .6, D=.1,df=30)

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

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

**Usage**

```r
multipleAlignment(pd, group, bw.gap=0.8, wn.gap=0.6, bw.D=.20, wn.D=.05, filterMin=3, lite=FALSE, usePeaks=TRUE, df=50, verbose=TRUE, timeAdjust=FALSE, doImpute=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
- `wn.D`: distance penalty for "within" alignments
- `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

**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


See Also

peaksDataset, betweenAlignment, progressiveAlignment

Examples

```r
require(gcspikelite)

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

cdfFiles <- dir(gcmsPath,"CDF",full=TRUE)
eluFiles <- dir(gcmsPath,"ELU",full=TRUE)

cdfFiles[1:2]

# 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=.05,bw.gap=0.6,bw.D=.2,usePeaks=TRUE,filterMin=1,df=50,verbose=TRUE)
```

### normDotProduct

Normalized Dot Product

**Description**

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

**Usage**

```r
normDotProduct(x1,x2,t1=NULL,t2=NULL,df=max(ncol(x1),ncol(x2)),D=100000,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.
parseChromaTOF

Value

matrix of similarities

Author(s)

Mark Robinson

References


See Also
dp, peaksAlignment

Examples

```r
require(gcspikelite)

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

# read data, peak detection results
pd<-peaksDataset(cddfFiles[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  Parser for ChromaTOF files

Description

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

Usage

```r
parseChromaTOF(fn,min.pc=.01,mz=seq(85,500),rt.cut=.008,rtrange=NULL,skip=1,rtDivide=60)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fn</td>
<td>ChromaTOF filename to read.</td>
</tr>
<tr>
<td>min.pc</td>
<td>minimum percent of maximum intensity.</td>
</tr>
<tr>
<td>mz</td>
<td>vector of mass-to-charge bins of raw data table.</td>
</tr>
<tr>
<td>rt.cut</td>
<td>the difference in retention time, below which peaks are merged together.</td>
</tr>
<tr>
<td>rtrange</td>
<td>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)</td>
</tr>
<tr>
<td>skip</td>
<td>number of rows to skip at beginning of the ChromaTOF</td>
</tr>
<tr>
<td>rtDivide</td>
<td>multiplier to divide the retention times by (default: 60)</td>
</tr>
</tbody>
</table>
parseELU

Description

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

Usage

```r
parseELU(f, min.pc = .01, mz = seq(50, 550), rt.cut = .008, rtrange = NULL)
```
**parseELU**

**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**


**See Also**

`addAMDISPeaks`

**Examples**

```r
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=.5,D=1000,timedf=NULL,df=30,verbose=TRUE,usePeaks=TRUE,compress=TRUE)

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
D time penalty (on same scale as retention time differences, t1 and t2)
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.

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

Value
peaksAlignment object

Author(s)
Mark Robinson

References

See Also
peaksDataset, clusterAlignment
peaksDataset

Examples

# see clusterAlignment, it calls peaksAlignment

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


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)
plot.peaksDataset  

Plotting functions for GCMS data objects

Description

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

Usage

```r
.plotpD(object, runs = 1:length(object@rawdata), mzind = 1:nrow(object@rawdata[[1]]),
        mind = NULL, plotSampleLabels = TRUE, calcGlobalMax = FALSE, peakCex = 0.8,
        plotPeakBoundaries = FALSE, plotPeakLabels = FALSE, plotMergedPeakLabels = TRUE, mlwd = 3,
        usePeaks = TRUE, plotAcrossRuns = FALSE, overlap = FALSE, rtrange = NULL, cols = NULL, thin = 1,
        max.near = median(object@rawrt[[1]]), how.near = 50, scale.up = 1, ...)

.plotpA(object, xlab = "Peaks - run 1", ylab = "Peaks - run 2", plotMatches = TRUE, matchPch = 19,
        matchLwd = 3, matchCex = 0.5, matchCol = "black", col = colorpanel(50, "black", "blue", "white"),
        breaks = seq(0, 1, length = 51), ...)

.plotcA(object, alignment = 1, ...)
```

Arguments

- `object`: a `peaksDataset`, `peaksAlignment` or `clusterAlignment` object.
- `runs`: for `peaksDataset` only: set of run indices to plot.
- `mzind`: for `peaksDataset` only: set of mass-to-charge indices to sum over (default, all).
- `mind`: for `peaksDataset` only: matrix of aligned indices.
- `plotSampleLabels`: for `peaksDataset` only: logical, whether to display sample labels.
- `calcGlobalMax`: for `peaksDataset` only: logical, whether to calculate an overall maximum for scaling.
- `peakCex`: character expansion factor for peak labels.
- `plotPeaks`: for `peaksDataset` only: logical, whether to plot hashes for each peak.
- `plotPeakBoundaries`: for `peaksDataset` only: logical, whether to display peak boundaries.
- `plotPeakLabels`: for `peaksDataset` only: logical, whether to display peak labels.
- `plotMergedPeakLabels`: for `peaksDataset` only: logical, whether to display 'merged' peak labels.
- `mlwd`: for `peaksDataset` only: line width of lines indicating the alignment.
- `usePeaks`: for `peaksDataset` only: logical, whether to plot alignment of peaks (otherwise, scans).
- `plotAcrossRuns`: for `peaksDataset` only: logical, whether to plot across peaks when unmatched peak is given.
- `overlap`: for `peaksDataset` only: logical, whether to plot TIC/XICs overlapping.
- `rtrange`: for `peaksDataset` only: vector of length 2 giving start and end of the X-axis.
- `cols`: for `peaksDataset` only: vector of colours (same length as the length of runs).
plot.peaksDataset

thin for peaksDataset only: when usePeaks=FALSE, plot the alignment lines every thin values
max.near for peaksDataset only: where to look for maximum
how.near for peaksDataset only: how far away from max.near to look
scale.up for peaksDataset only: a constant factor to scale the TICs
plotMatches for peaksDataset only: logical, whether to plot matches
xlab for peaksAlignment and clusterAlignment only: x-axis label
ylab for peaksAlignment and clusterAlignment only: y-axis label
matchPch for peaksAlignment and clusterAlignment only: match plotting character
matchLwd for peaksAlignment and clusterAlignment only: match line width
matchCex for peaksAlignment and clusterAlignment only: match character expansion factor
matchCol for peaksAlignment and clusterAlignment only: match colour
col for peaksAlignment and clusterAlignment only: vector of colours for colourscale
breaks for peaksAlignment and clusterAlignment only: vector of breaks for colourscale
alignment for peaksAlignment and clusterAlignment only: the set of alignments to plot

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


See Also

plotImage, 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),rtrange=c(7.5,8.5))
# image plot
plot(pd, rtrange=c(7.5, 8.5), plotPeaks=TRUE, plotPeakLabels=TRUE)

## plotImage

### Plot of images of GCMS data

#### Description

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

#### Usage

```r
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 scaled to the maximum value (as specified by the `how.near` and `max.near` values). The many parameters give considerable flexibility on 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


#### See Also

- `plot`, `peaksDataset`
Examples

```r
require(gcspikelite)
# paths and files
gcmsPath<-paste(find.package("gcspikelite"),"data",sep="/"
CDF"
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="")
```

Description

Plot the mass spectra from the profile matrix

Usage

```r
plotSpectra(object, sample, spectraID, 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
- **spectraID**: 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 other peaks are scaled consequetially
- **...**: other parameter passed to the plot() function

Details

Plot the deconvoluted mass spectra from the profile matrix

Author(s)

riccardo.romoli@unifi.it

Examples

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

```r
## full paths to file names
cdfFiles <- dir(gcmsPath, "CDF", full=TRUE)
```
## create a `peaksDataset` object and add XCMS peaks to it
```r
pd <- peaksDataset(cdfFiles[1:3], mz=seq(50,550), rtrange=c(7.5,8.5))
pd <- addXCMSPeaks(cdfFiles[1:3], pd, peakPicking=c('mf'), snthresh=5,
                   fwhm=10, step=1, steps=2, mzdif=0.5, sleep=0)
```

## align the chromatograms
```r
mp <- correlationAlignment(object=pd, thr=0.8, D=20,
                           penality=0.2, normalize=TRUE, minFilter=2)
```

## view the alignment results
```r
mp@Alignment
```

## plot the mass spectra
```r
par(mfrow=c(3,1))
plotSpectra(object=pd, sample=cdfFiles[1], spectraID=2)
plotSpectra(object=pd, sample=cdfFiles[2], spectraID=3)
plotSpectra(object=pd, sample=cdfFiles[3], spectraID=4)
```

---

### 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**

```r
progressiveAlignment(pD, cA, D=1000, gap=.5, verbose=TRUE, usePeaks=TRUE, df=30, compress=TRUE)
```

**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

**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
### Description

Build a fat data matrix

### Usage

```r
retFatMatrix(data)
```

### Arguments

- **data**: the list obtained from `gatherInfo()`

### 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>

See Also

gatherInfo

Examples

```
require(gcspikelite)
# paths and files
gcmsPath <- paste(find.package("gcspikelite"), "data", sep="/"

cdfFiles <- dir(gcmsPath,"CDF",full=TRUE)
# read data, peak detection results
pd <- peaksDataset(cdfFiles[1:2], mz=seq(50,550),
                   rtrange=c(7.5,8.5))

pd <- addXCMSPeaks(files=cdfFiles[1:2], object=pd,
                   peakPicking="mF", snthresh=3, fwhm=4,
                   step=1, steps=2, mzdff=0.5)

mp <- correlationAlignment(object=pd, thr=0.85, D=20,
                           penalty=0.2, normalize=TRUE,
                           minfilter=1)

outList <- gatherInfo(pd, mp)

mtxD <- retFatMatrix(data=outList)
```

---

### rmaFitUnit

R fits a robust linear model (RLM) for one metabolite

**Description**

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

**Usage**

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

**Arguments**

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

**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


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))

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

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

*Topic classes
betweenAlignment, 5
clusterAlignment, 7
correlationAlignment, 10
correlationAlignment-class, 11
eitherMatrix-class, 13
multipleAlignment-class, 17
peaksAlignment-class, 22
peaksDataset, 23
plot.peaksDataset, 24
plotImage, 26
progressiveAlignment-class, 28

*Topic manip
addAMDISPeaks, 2
addChromaTOFPeaks, 3
addXCMSPeaks, 4
calcTimeDiffs, 6
compress, 9
dp, 12
gatherInfo, 14
imputePeaks, 15
normDotProduct, 18
parseChromaTOF, 19
parseELU, 20
rmFitUnit, 30
.plotcA (plot.peaksDataset), 24
.plotpA (plot.peaksDataset), 24
.plotpD (plot.peaksDataset), 24
addAMDISPeaks, 2, 20, 21
addChromaTOFPeaks, 3, 20
addXCMSPeaks, 4, 10
align, correlationAlignment
 (correlationAlignment-class), 11
betweenAlignment, 5, 18
betweenAlignment-class
 (betweenAlignment), 5
betweenAlignment-show
 (betweenAlignment), 5
calcTimeDiffs, 6
correlationAlignment, 7, 7, 9, 22, 31
correlationAlignment-class
 (clusterAlignment), 7
correlationAlignment-plot
 (clusterAlignment), 7
correlationAlignment-show
 (clusterAlignment), 7
correlationAlignment-class
 (betweenAlignment), 10
correlationAlignment-class
 (clusterAlignment), 7
correlationAlignment-class
 (peaksAlignment-class), 22
correlationAlignment-class
 (peaksDataset), 23
correlationAlignment-class
 (plot.peaksDataset), 24
correlationAlignment-class
 (plotImage), 26
correlationAlignment-class
 (progressiveAlignment-class), 28
correlationAlignment-class
 (betweenAlignment), 10
correlationAlignment-class
 (clusterAlignment), 11
correlationAlignment-class
 (eitherMatrix-class), 12
correlationAlignment-class
 (multipleAlignment-class), 17
correlationAlignment-class
 (multipleAlignment-class), 17
correlationAlignment-class
 (multipleAlignment-class), 17
correlationAlignment-class
 (peaksAlignment-class), 22
compress, 9
compress, clusterAlignment-method
 (compress), 9
compress, peaksAlignment-method
 (compress), 9
compress, progressiveAlignment-method
 (compress), 9
correrlationAlignment, 10, 11
correrlationAlignment-class, 11
correrlationAlignment-show
 (correlationAlignment-class), 11
decompress (compress), 9
decompress, clusterAlignment-method
 (compress), 9
decompress, peaksAlignment-method
 (compress), 9
decompress, progressiveAlignment-method
 (compress), 9
dp, 12, 19
eitherMatrix-class, 13
exportSpectra, 13
findPeaks.centWave, 5
findPeaks.matchedFilter, 5
gatherInfo, 14, 30
imputePeaks, 15, 15
multipleAlignment, 6, 16, 29
multipleAlignment
 (multipleAlignment-class), 17
multipleAlignment-class, 17
multipleAlignment-class
 (multipleAlignment-class), 17
multipleAlignment-class
 (multipleAlignment-class), 17
INDEX

normDotProduct, 12, 18
parseChromaTOF, 4, 19
parseELU, 3, 20
peaksAlignment, 7–9, 13, 19-31
peaksAlignment (peaksAlignment-class), 22
peaksAlignment-class, 22
peaksAlignment-plot (peaksAlignment-class), 22
peaksAlignment-show (peaksAlignment-class), 22
peaksDataset, 3–5, 8, 10, 16, 18, 22, 23, 25, 26, 29
peaksDataset-class (peaksDataset), 23
peaksDataset-plot (peaksDataset), 23
peaksDataset-show (peaksDataset), 23
plot, 26
plot (plot.peaksDataset), 24
plot,clusterAlignment,ANY-method (clusterAlignment), 7
plot,clusterAlignment-method (clusterAlignment), 7
plot,peaksAlignment,ANY-method (peaksAlignment-class), 22
plot,peaksAlignment-method (peaksAlignment-class), 22
plot,peaksDataset,ANY-method (peaksDataset), 23
plot,peaksDataset-method (peaksDataset), 23
plot.peaksDataset, 24
plotImage, 25, 26
plotImage,peaksDataset-method (plotImage), 26
plotSpectra, 27
progressiveAlignment, 9, 16, 18
progressiveAlignment (progressiveAlignment-class), 28
progressiveAlignment-class, 28
progressiveAlignment-show (progressiveAlignment-class), 28
retFatMatrix, 29
rmaFitUnit, 30
show,betweenAlignment-method (betweenAlignment), 5
show,clusterAlignment-method (clusterAlignment), 7
show,correlationAlignment-method (correlationAlignment-class), 11
show,correlationAlignment-method, ANY-methods (correlationAlignment-class), 11
show,multipleAlignment-method (multipleAlignment-class), 17
show,peaksAlignment-method (peaksAlignment-class), 22
show,peaksDataset-method (peaksDataset), 23
show,progressiveAlignment-method (progressiveAlignment-class), 28