**flagme**

November 11, 2009

### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>addAMDISPeaks</td>
<td>1</td>
</tr>
<tr>
<td>addChromaTOFPeaks</td>
<td>2</td>
</tr>
<tr>
<td>betweenAlignment</td>
<td>4</td>
</tr>
<tr>
<td>calcTimeDiffs</td>
<td>5</td>
</tr>
<tr>
<td>clusterAlignment</td>
<td>6</td>
</tr>
<tr>
<td>compress</td>
<td>7</td>
</tr>
<tr>
<td>dp</td>
<td>8</td>
</tr>
<tr>
<td>gatherInfo</td>
<td>10</td>
</tr>
<tr>
<td>imputePeaks</td>
<td>11</td>
</tr>
<tr>
<td>multipleAlignment-class</td>
<td>13</td>
</tr>
<tr>
<td>normDotProduct</td>
<td>14</td>
</tr>
<tr>
<td>parseChromaTOF</td>
<td>15</td>
</tr>
<tr>
<td>parseELU</td>
<td>16</td>
</tr>
<tr>
<td>peaksAlignment-class</td>
<td>18</td>
</tr>
<tr>
<td>peaksDataset</td>
<td>19</td>
</tr>
<tr>
<td>plotImage</td>
<td>20</td>
</tr>
<tr>
<td>plot.peaksDataset</td>
<td>21</td>
</tr>
<tr>
<td>progressiveAlignment-class</td>
<td>23</td>
</tr>
<tr>
<td>rmaFitUnit</td>
<td>24</td>
</tr>
</tbody>
</table>

**Index** 26

### addAMDISPeaks

*Add AMDIS peak detection results*

**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

`parseELU`, `peaksDataset`

### Examples

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

data <- gcmsPath<-
```

```r
# 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])
```

---

### Description

Adds ChromaTOF peak detection results.

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

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
...

Details

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

Value

peaksDataset object

Author(s)

Mark Robinson

References


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,...)
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)

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
calcTimeDiffs

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,

**Value**

- A `list` of the 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="/")
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**  
*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=)
```

**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
compress

Author(s)

Mark Robinson

References


See Also

peaksDataset, peaksAlignment

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])

c<-clusterAlignment(pd, gap = .5,D=.05,df=30)
```

compress  

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 `SpireM` package.
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=.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

A list with elements `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="/")
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)
```
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 routine 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

Author(s)

Mark Robinson

References

**imputePeaks**  
**Imputation 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**

```r
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

- `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 procedure 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

```
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 = .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

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

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 tgs$Group label with 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
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=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 = "/")
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**  
**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**

- `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)
parseELU

Details

parseChromaTOF will typically be called by \texttt{addChromaTOFPeaks}, not called directly.

Peaks that are detected within \texttt{rt.cut} are merged together. This avoids peaks which are essentially overlapping.

Fragments that are less than \texttt{min.pc} of the maximum intensity fragment are discarded.

Value

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

Author(s)

Mark Robinson

References


See Also

\texttt{addAMDISPeaks}

Examples

require(gcspikelite)

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

# parse ChromaTOF file
cTofList<-parseChromaTOF(tofFiles[1])
\end{verbatim}

\begin{verbatim}
parseELU  Parser for ELU files
\end{verbatim}

Description

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

Usage

\begin{verbatim}
parseELU(f,min.pc=.01,mz=seq(50,550),rt.cut=.008,rtrange=NULL)
\end{verbatim}
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
peaksDataset

See Also

peaksDataset, clusterAlignment

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)

plotImage

Plot of images of GCMS data

Description

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

Usage

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

**plot.peaksDataset**

**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=F, 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=.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

**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
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="")
```

See Also

`plot.peaksDataset`
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)
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
...
... further arguments passed to the plot or 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.
progressiveAlignment-class

Author(s)

Mark Robinson

References


See Also

plotImage, peaksDataset

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
df<-peaksDataset(cdfFiles[1:3],mz=seq(50,550),rtrange=c(7.5,8.5))

# image plot
plot(df,rtrange=c(7.5,8.5),plotPeaks=TRUE,plotPeakLabels=TRUE)
```

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

Author(s)

Mark Robinson

References


See Also

peaksDataset, multipleAlignment

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

rmaFitUnit

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

rmaFitUnit(u,maxit=5,mzEffect=TRUE,cls=NULL,fitSample=TRUE,fitOrCoef=c("coef","
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 fragmentxsample intensity 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

```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])

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

timedf<-calcTimeDiffs(pd, fullca)
```
Index

**Topic classes**
- betweenAlignment, 3
- clusterAlignment, 5
- multipleAlignment-class, 12
- peaksAlignment-class, 17
- peaksDataset, 18
- plot.peaksDataset, 20
- plotImage, 19
- progressiveAlignment-class, 22

**Topic manip**
- addAMDISPeaks, 1
- addChromaTOFPeaks, 2
- calcTimeDiffs, 4
- compress, 7
- dp, 8
- gatherInfo, 9
- imputePeaks, 10
- normDotProduct, 13
- parseChromaTOF, 14
- parseELU, 16
- rmaFitUnit, 24
- .plotcA(plot.peaksDataset), 20
- .plotpA(plot.peaksDataset), 20
- .plotpD(plot.peaksDataset), 20

addAMDISPeaks, 1, 15, 16
addChromaTOFPeaks, 2, 15
betweenAlignment, 3, 13
betweenAlignment-class
  (betweenAlignment), 3
betweenAlignment-show
  (betweenAlignment), 3
calcTimeDiffs, 4
clusterAlignment, 5, 5, 7, 18, 24
clusterAlignment-class
  (clusterAlignment), 5
clusterAlignment-plot
  (clusterAlignment), 5
clusterAlignment-show
  (clusterAlignment), 5
compress, 7
compress, clusterAlignment-method
  (compress), 7
compress, peaksAlignment-method
  (compress), 7
compress, progressiveAlignment-method
  (compress), 7
decompress(compress), 7
decompress, clusterAlignment-method
  (compress), 7
decompress, peaksAlignment-method
  (compress), 7
decompress, progressiveAlignment-method
  (compress), 7
dp, 8, 14
gatherInfo, 9
imputePeaks, 10, 10
multipleAlignment, 4, 11, 23
multipleAlignment
  (multipleAlignment-class), 12
multipleAlignment-class, 12
multipleAlignment-show
  (multipleAlignment-class), 12
normDotProduct, 8, 13
parseChromaTOF, 3, 14
parseELU, 2, 16
peaksAlignment, 5–7, 14, 24
peaksAlignment
  (peaksAlignment-class), 17
peaksAlignment-class, 17
peaksAlignment-plot
  (peaksAlignment-class), 17
peaksAlignment-show
  (peaksAlignment-class), 17
peaksDataset, 2, 3, 6, 11, 13, 18, 18, 20, 22, 23
peaksDataset-class
  (peaksDataset), 18
peaksDataset-plot (peaksDataset), 18
peaksDataset-show (peaksDataset), 18
plot, 20
plot (plot.peaksDataset), 20
plot, clusterAlignment-method  
  (clusterAlignment), 5
plot, peaksAlignment-method  
  (peaksAlignment-class), 17
plot, peaksDataset-method  
  (peaksDataset), 18
plot.peaksDataset, 20
plotImage, 19, 22
plotImage, peaksDataset-method  
  (plotImage), 19
progressiveAlignment, 7, 11, 13
progressiveAlignment  
  (progressiveAlignment-class), 22
progressiveAlignment-class, 22
progressiveAlignment-show  
  (progressiveAlignment-class), 22
rmaFitUnit, 24
show, betweenAlignment-method  
  (betweenAlignment), 3
show, clusterAlignment-method  
  (clusterAlignment), 5
show, multipleAlignment-method  
  (multipleAlignment-class), 12
show, peaksAlignment-method  
  (peaksAlignment-class), 17
show, peaksDataset-method  
  (peaksDataset), 18
show, progressiveAlignment-method  
  (progressiveAlignment-class), 22