Package 'IPPD'

April 16, 2019

Title Isotopic peak pattern deconvolution for Protein Mass Spectrometry by template matching

Version 1.30.0

Date 2014-07-24

Author Martin Slawski <ms@cs.uni-saarland.de>, Rene Hussong <rene.hussong@uni.lu>, Andreas Hildebrandt <andreas.hildebrandt@uni-mainz.de>, Matthias Hein <hein@cs.uni-saarland.de>

Maintainer Martin Slawski <ms@cs.uni-saarland.de>

Depends R (>= 2.12.0), MASS, Matrix, XML, digest, bitops

Imports methods, stats, graphics

Description The package provides functionality to extract isotopic peak patterns from raw mass spectra. This is done by fitting a large set of template basis functions to the raw spectrum using either nonnegative least squares or least absolute deviation fittting. The package offers a flexible function which tries to estimate model parameters in a way tailored to the peak shapes in the data. The package also provides functionality to process LCMS runs.

License GPL (version 2 or later)

Collate classes.R methods.R getPeaklist.R fitModelParameters.R internals.R analyzeLCMS.R read.mzXML.R sweepline.R

biocViews Proteomics

git_url https://git.bioconductor.org/packages/IPPD

git_branch RELEASE_3_8

git_last_commit 6aea9f2

git_last_commit_date 2018-10-30

Date/Publication 2019-04-15

R topics documented:

`IPPD-package` *Peak pattern deconvolution for Protein Mass Spectrometry by non-negative ls/lad template matching*

Description

The package provides functionality to extract isotopic peak patterns from raw mass spectra. This is done by fitting a large set of template basis functions to the raw spectrum using nonnegative least squares (ls) or nonnegative least absolute deviation (lad). Ideally, the nonnegativity constraint in combination with nonnegativity of the template basis functions effects that templates not matching the data are assigned an extremely low weight such that one can easily identify isotopic patterns present and not present in the spectrum. In practice, the templates only approximate the peak patterns, where the quality the approximation crucially depends on how well the shapes of the templates fit the isotopic patterns contained in a spectrum. For this reason, the package offers the flexible function fitModelParameters which tries to estimate model parameters, e.g. the width of a Gaussian bump, in a way tailored to the peak shapes in the data. As second peak model in addition to the standard Gaussian, the package offers full support for the Exponential Modified Gaussian.
The function getPeaklist predicts the set of isotopic peak patterns present in the spectrum in a fully automatic, yet customizable way. The main benefits of our approach are that

1. Overlapping peak patterns can be resolved.
2. The complete spectrum can be processed as a whole or in large sections by exploiting the sparse nature of the problem.
3. The set of parameters in getPeaklist are easy to interpret and require only very basic knowledge of statistics.
4. A theoretically well-founded post-processing procedure is used.
5. The result can be analyzed visually in a detailed way using the function `visualize`.

Package:	IPPD
Type:	Package
Version:	1.3.1
Date:	2012-01-17
License:	GPL (version 2 or later)

Author(s)

Martin Slawski <ms@cs.uni-saarland.de>,
Rene Hussong <rene.hussong@uni.lu>,
Andreas Hildebrandt <andreas.hildebrandt@uni-mainz.de>,
Matthias Hein <hein@cs.uni-saarland.de>

Maintainer: Martin Slawski <ms@cs.uni-saarland.de>.

analyzeLCMS *Functionality for LC-MS data*

Description

The function analyzeLCMS applies the function getPeaklist to all scans of the dataset. The sweep line scheme of Schulz-Trieglaff et al. (2008) is then used to find retention time intervals of consecutive signals.

Usage

```
analyzeLCMS(data, arglist.fitModelParameters = list(),
                        arglist.getPeaklist = list(),
                        arglist.threshold = list(),
                        arglist.sweepline = list(),
                        trace = TRUE)
```

Arguments

data	data can be one of the following: • A character for the path to the file that is to be processed. The file has to be either an mzXML file or a three-column matrix whose columns contains retention times, m/z positions and intensities (exactly in that order). • An object of class mzXML. • A matrix or data.frame containing three columns containing retention times, m/z positions and intensities (exactly in that order).
arglist.fitModelParameters	Optional arguments to be passed to fitModelParameters.
arglist.getPeaklist	Optional arguments to be passed to getPeaklist.
arglist.threshold	Optional arguments to be passed to the method threshold, see peaklist for details.
arglist.sweepline	Optional arguments to be passed to the function sweepline.
trace	A logical controlling whether status information is displayed.

Value

A list composed of

peaklists A list of matrices as in the slot peaklistprocessed of an object of class peaklist. These matrices are obtained from running getPeaklist for all scans contained in data and applying hard thresholding as implemented in the method threshold.

boxes A matrix of retention time intervals as returned as output by a call to the function sweepline.

References

0. Schulz-Trieglaff and R. Hussong and C. Groepl and A. Leinenbach and A. Hildebrandt and C. Huber and K. Reinert. (2008) *Journal of Computational Biology, 15, 685-704*

`fitModelParameters` *Peak parameter estimation*

Description

In the template-based approach of this package, each template/peak pattern is composed of several single basic peaks. The shape of such a basic peak may be modeled either as Gaussian or as Exponentially Modified Gaussian (EMG). The second model assumes that the shape of each peak equals the shape of the function one obtains when convolving the probability density function of a Gaussian distribution with the probability density function of the exponential distribution. This is a more complex as well as more flexible model, since it allows one to account for skewness. Both peak models depend on a set of parameters which are usually unknown a priori. Moreover, these parameters tend to vary over the spectrum. The documented method provides the following functionality: given a raw spectrum and a linear model that describes how the set of parameters varies as a function of m/z, we detect well-resolved peaks within the spectrum. For each detected peak, we determine the parameters of our model in such a way that the resulting peak shape matches that of the detected peak as good as possible in a least-squares sense. In this manner, we obtain a set of estimated parameters for different m/z positions. These estimates are used as response, the m/z positions as explanatory variable in a linear regression model (not necessarily linear in m/z). To be resistant to outliers (e.g. as occurring due to overlapping peaks), we use least absolute deviation regression to infer the model parameters of these linear models. The result can directly be used for the argument `model.parameters` in the main method `getPeakList`.

Usage

```
fitModelParameters(mz, intensities, model = c("Gaussian", "EMG"),
                                            fitting = c("most_intense", "model"),
                                             formula.alpha =  formula(~1),
                                             formula.sigma = formula(~1),
                                             formula.mu = formula(~1),
                                             control = list(window = 6,
                                             threshold = NULL, rlm.maxit = 20))
```

Arguments

`mz`	A `numeric` vector of m/z (mass/charge) values (in Thomson).
`intensities`	A `numeric` vector of intensities corresponding to `mz`.
`model`	Basic model for the shape of a single peak. Must be `"Gaussian"` or `"EMG"` (exponentially modified Gaussian). See details below.
`fitting`	A character specifying the mode of peak extraction and -fitting. If `fitting = "most_intense"`, then the most intense peak is detected and the parameter(s) is (are) fitted using only this peak. Then, the resulting functions of the form `parameter(mz)` are all constant functions. If `fitting = "model"`, as many peaks as possible satisfying the criteria of `control` are used to estimate linear models of the form `parameter(mz) = beta_0 + beta_1 g_1(mz) + ... + beta_p g_p(mz)`, where the g's are fixed functions. Their specification is performed by specifying one-sided formulae according to the Wilkinson-Roger notation for linear models as in the function `lm`. The model formulae have to satisfy the following criteria:

- The formula is one sided, i.e. *no* term appears on the left hand side of `~`.

- The right hand side consists only of functions in mz, and mz is the only variable that may be used. Product terms involving * are *not* admitted.
- **Important**: Note that, for example ~ 1 + mz + sqrt(mz) is a valid formula in the sense that no error will occur, but it does *not* correspond to the linear model parameter(mz) = beta_0 + beta_1 mz + beta_2 sqrt(mz). The correct model formula instead reads ~ 1 + mz + I(sqrt(mz)), i.e. each function has to be bracketed by I().

formula.alpha A one-sided formula describing the dependence of the EMG-parameter alpha as a function of m/z, s. fitting. The default assumes that the parameter is independent of m/z, in which case one obtains a function returning the median of the estimates obtained for different detected peaks. formula.alpha, formula.sigma and formula.mu are needed if and only if fitting = "model".

formula.sigma Parameter used for both peak models. For further information, see formula.alpha

formula.mu See formula.alpha. Used only if model = "EMG".

control A list controlling peak detection. The parameter window refers to the minimal resolution of a peak. According to window, a sequence of intensities at adjacent m/z positions is considered as peak if and only if there are at least window m/z positions with *increasing* intensity followed by a second sequence of window m/z positions with *decreasing* intensity. If in addition threshold is specified, only peaks whose maximum intensity is equal to or greater than threshold is considered. **Note**: Usually, threshold is specified, since otherwise the maximum intensity among the complete spectrum minus some epsilon is taken as threshold. rlm.maxit allows to control the maximum number of iterations used to fit peaks to the data.

Details

Let the variable x represent m/z. Then model = "Gaussian" assumes that a single peak can be described as

```
gaussfun(x;sigma,mu) = exp(-(x - mu)^2/sigma)
```

The parameter mu is *not* considered as model parameter: in the computation of the resulting basis function matrix, mu is always set to a known m/z position where the leading peak of a peak pattern might be present.
Model = "EMG" assumes that a single peak can be described as

```
EMG(x;alpha,sigma,mu) = exp(sigma^2/(2 * alpha^2) + (mu -  x)/alpha) (1 - Phi(sigma/alpha + (mu -
```

where Phi represents the cumulative density function of the standard Gaussian distribution. Alternatively, EMG(.;alpha,sigma,mu) can be expressed as
EMG(x;alpha,sigma,mu) = (phi ** gamma)(x),
where ** denotes convolution, phi is the density function of the Gaussian distribution with mean mu and standard deviation sigma and gamma is the density function of an exponential distribution with expectation alpha.
The parameters of EMG can be interpreted as follows.

alpha The lower alpha, the more the shape of the peak resembles that of a Gaussian. Conversely, large values of alpha lead to long right tails.

sigma Controls the width of the peak (together with alpha).

mu A location parameter. Note that in general mu does *not* coincide with the mode of EMG. Therefore, if model = "EMG", all three parameters are estimated from detected peaks.

Moreover, the skewness of EMG is characterized by the ratio alpha/sigma.

Value

An object of class modelfit.

Warning

Parameter estimation by fitting detected peaks is possible only if single peaks are sufficiently well-resolved. A peak composed of, say, five (m/z, intensity) pairs, is inappropriate to infer three parameters.

Warning

The choice model = "EMG" in conjunction with fitting = "model" can be extremely slow (taking up to several minutes of computing time) if many peaks are detected and fitted. This is caused by a grid search over a grid of 10^6 different combinations of alpha, sigma and mu performed prior to nonlinear least squares estimation in order to find suitable starting values.

See Also

getPeaklist, modelfit

Examples

```
### load data
data(toyspectrum)
### estimate parameter sigma of a Gaussian model,
### assumed to be independent of m/z

simplegauss <- fitModelParameters(toyspectrum[,1],
             toyspectrum[,2],
             model = "Gaussian",
             fitting = c("model"),
             formula.sigma = formula(~1),
             control = list(window = 6, threshold = 1))

show(simplegauss)
visualize(simplegauss, type = "peak", xlab = "m/z", ylab = "intensity",
          main = "Gaussian fit")

### fit the model sigma(m/z) = beta_0 + beta_1 m/z + beta_2 m/z^2

gaussquadratic <- fitModelParameters(toyspectrum[,1],
             toyspectrum[,2],
             model = "Gaussian",
             fitting = "model",
             formula.sigma = formula(~mz + I(mz^2) ),
             control = list(window = 6, threshold = 1))

show(gaussquadratic)
visualize(gaussquadratic, type = "model", modelfit = TRUE)

### estimate parameters for EMG-shaped peaks

EMGlinear <- fitModelParameters(toyspectrum[,1],
```

```
              toyspectrum[,2],
              model = "EMG",
              fitting = "model",
              formula.alpha = formula(~mz),
              formula.sigma = formula(~mz),
              formula.mu = formula(~1),
              control = list(window = 6, threshold = 1))

show(EMGlinear)

visualize(EMGlinear, type = "peak", xlab = "m/z", ylab = "intensities",
          main = "EMG fit")

visualize(EMGlinear, type = "model", parameters = c("alpha", "sigma"), modelfit = TRUE)
```

`getPeaklist` *Peak pattern extraction by non-negative ls/lad template matching*

Description

Generates a candidate list of isotopic peak patterns present in a protein mass spectrum. This is achieved by matching templates calculated according to the so-called Averagine model to the raw spectrum using either non-negative least squares (ls) or non-negative least absolute deviation (lad) estimation. The presence of multiple charge states is supported. In particular, the approach is capable of deconvolving overlapping patterns. The method can be applied with two different kind of peak shapes, Gaussians and Exponentially Modified Gaussians (EMG).

Usage

```
getPeaklist(mz, intensities, model = c("Gaussian", "EMG"),
model.parameters = list(alpha = function(){},
                        sigma = function(){},
                        mu = function(){}),
averagine.table = NULL,
loss = c("L2", "L1"), binning = FALSE,
postprocessing = TRUE, trace = TRUE, returnbasis = TRUE,
control.basis = list(charges = c(1,2,3,4),
                     eps = 1e-05),
control.localnoise = list(quantile = 0.5,
                          factor.place = 1.5,
                          factor.post = 0,
                          window = NULL,
                          subtract = FALSE),
control.postprocessing = list(mzfilter = FALSE,
                              prune = FALSE,
                              factor.prune = NULL,
                              ppm = NULL,
                              goodnessoffit = FALSE),
control.binning = list(tol = 0.01))
```

Arguments

mz	A numeric vector of m/z (mass/charge) values (in Thomson), ordered increasingly.
intensities	A numeric vector of intensities corresponding to mz.
model	Basic model for the shape of a single peak. Must be "Gaussian" or "EMG" (exponentially modified Gaussian). See fitModelParameters for further information on these models.
averagine.table	If averagine.table = NULL the usual Averagine model (Senko et al. (1995): Determination of Monoisotopic Masses and Ion Populations for Large Biomolecules from Resolved Isotopic Distributions, J. Am. Soc. Mass Spect., 6, 229-233) is used. Otherwise, avergine.table has to be a matrix or data.frame of the following form: each row encodes the isotope distribution at a certain mass. The first entry of each row contains such mass while the remaining entries contain the relative abundances of the isotopes.
loss	The loss function to be used. The choice loss = "L2" yield a nonnegative least squares fit, loss = "L1" a nonnegative least absolute deviation fit. The second choice is more robust when deviations from model assumptions (peak model, Averagine model,...) frequently occur in the data. Note, however, that computation time is much higher for the second choice (at least by a factor two).
model.parameters	A list of functions with precisely one argument representing mz. The parameters of a single peak are typically modeled as a function of m/z. If model = "Gaussian", the peak shape depends on the parameter sigma (a function sigma(mz)). If model = "EMG", the peak shape additionally depends on two parameters alpha and mu (two functions alpha(mz) and mu(mz)). Note that constant functions are usually specified by using a construction of the form parameter(mz) <- function(mz) rep(constant, length(mz)). Moreover, note that a valid function has to be vectorized. For the automatic generation of those functions from a raw spectrum and further details on the meaning of the parameters, see fitModelParameters. The output of a call to fitModelParameters can directly be plugged into getPeaklist via the argument model.parameters.
binning	A logical indicating whether the fitting process should be done sequentially in 'bins'. If TRUE, the spectrum is cut into pieces (bins). Each bin is then fitted separately, and the results of all bins are combined in the end. Division into bins may be configured using control.binning. See also the 'Details' section below.
postprocessing	A logical indicating whether a post-processing correction should be applied to the raw peaklist. See also the argument control.postprocessing and the 'Details' section below.
trace	A logical indicating whether information tracing the different steps of the fitting process should be displayed.
returnbasis	A logical indicating whether the matrix of basis functions (template functions evaluated at mz) should be returned. Note that this may be expensive in terms of storage.
control.basis	A list of arguments controlling the computation of the matrix of basis functions: charges The set of charge states present in the spectrum. eps Function values below eps are set equal to precisely zero in order to make the basis function matrix sparse.

`control.localnoise`

A list of arguments controlling the placement and selection of basis functions on the basis of a 'local noise level':

`quantile` A value from `0.1, 0.2, ..., 0.9`, specifying the quantile of the intensities residing in a sliding m/z `window` (s. below) to be used as 'local noise level'.

`factor.place` Controls the placement of basis functions. A basis function is placed at an element of `mz` if and only if the intensity at that position exceeds the 'local noise level' by a factor at least equal to `factor.place`.

`factor.post` Controls which basis functions *enter* the postprocessing step. A basis function is discarded before the postprocessing step if its estimated amplitude does not exceed the `factor.post` times the 'local noise level'. By default `factor.post = 0`. The pre-filtering step before postprocessing is mainly done for computational speed-up, and `factor.post = 0` is supposed to yield the qualitatively best solution, though it may take additional time.

`window` The length of the sliding window used to compute `quantile`, to be specified in Thomson. By default, `window` is chosen in such a way that it equals the length of the support of an 'average' basis function for charge state one.

`subtract` A `logical` indicating whether the 'local noise level' should be subtracted from the observed intensities. Setting `subtract = TRUE` is typically beneficial in the sense that fitting of noise is reduced.

`control.postprocessing`

A list of arguments controlling the postprocessing step (provided `postprocessing = TRUE`):

`mzfilter` Setting `mzfilter = TRUE` removes basis functions at positions where peak patterns are highly improbable to occur, thereby removing peaks from the list which are likely to be noise peaks. This filter is sometimes called 'peptide mass rule', see *Zubarev et al. (1996): Accuracy Requirements for Peptide Characterization by Monoisotopic Molecular Measurements, Anal. Chem.,88,4060-4063*.

`prune`**,** `factor.prune` Setting `prune = TRUE` activates a crude scheme that removes low-intensity peaks (likely to be noise peaks), as frequently occurring in regions with extremely intense peaks. According to this scheme, a peak is removed from the peak list if its amplitude is less than `factor.prune` times the locally most intense amplitude, where `factor.prune` typically ranges from `0.01` to `0.1`.

`ppm` A ppm (= parts per million) tolerance value within which basis functions at different m/z positions are considered to be merged, s. 'Details' below. By default, that value is computed from the spacing of the first two m/z positions.

`goodnessoffit` A `logical` indicating whether a local goodness-of-fit adjustment of the signa-to-noise ratio should be computed. Yields usually more reliable evaluation of the detected patterns, but is computationally more demanding.

`control.binning`

Controls the division of the spectrum into bins (if `binning = TRUE`). Based on the 'local noise level' described in `control.localnoise`, if within a range of `(1+tol)` Thomson no further significant position occurs, a bin is closed, and a new one is not opened as long as a new significant position occurs..

Details

- While setting `binning = TRUE` yields a procedure which is less memory consuming than fitting the whole spectrum simultaneously (`binning = FALSE`), it may be inferior from a quality aspect, since division into bins has to be done with care. Otherwise, peak patterns might be split up into different bins, which would result into erroneous fitting.
- Postprocessing of the raw list usually yields a considerably improved result by counteracting the 'peak-splitting phenomenon': due to a limited sampling rate and discrepancies between theoretical and observed peak patterns, several templates at adjacent positions are used to fit the same peak pattern.

Value

An object of class peaklist.

Warning

Although we have tried to choose default values expected to produce sensible results, the user should carefully examine all options.

Warning

Depending on the length and the resolution of the raw spectrum, fitting the whole spectrum simultaneously as recommended is expensive from a computational point of view, and may take up to several minutes per spectrum.

See Also

fitModelParameters, peaklist

Examples

```
### load data
data(toyspectrum)
data(toyspectrumsolution)
mz <- toyspectrum[,"x"]
intensities <- toyspectrum[,"yyy"]
### select mz range
filter <- mz >= 2800 & mz <= 3200
### Extract peak patterns with model = "Gaussian"
sigmafun <- function (mz)  -8.5e-07 * mz + 6.09e-10 * mz^2 + 0.00076
gausslist <- getPeaklist(mz = mz[filter], intensities = intensities[filter],
                  model = "Gaussian",
                  model.parameters = list(sigma = sigmafun,
                                          alpha = function(mz){},
                                          mu = function(mz){}),
                   control.localnoise = list(quantile = 0.5, factor.place = 3))

show(gausslist)
### threshold list at signal-to-noise ratio = 2
peaklist <- threshold(gausslist, threshold = 2)

### Extract peak patterns with model = "EMG" and loss = "L1"
alpha0 <- function(mz) 0.00001875 * 0.5 * 4/3 * mz
sigma0 <- function(mz) 0.00001875 * 0.5 * mz
mu0 <- function(mz) return(rep(-0.06162891, length(mz)))
```

```
EMGlist <- getPeaklist(mz = mz[filter], intensities = intensities[filter],
                  model = "EMG", loss = "L1",
                  model.parameters = list(sigma = sigma0,
                                          alpha = alpha0,
                                          mu = mu0),
                  control.localnoise = list(quantile = 0.5, factor.place = 3))
show(EMGlist)
peaklist2 <- threshold(EMGlist, threshold = 2)

### plot results of the 1st list and compare vs. 'truth'

### 'ground truth'
solution <- toyspectrumsolution[toyspectrumsolution[,1] >= 2800 & toyspectrumsolution[,1] <= 3200,]

visualize(gausslist, mz[filter], intensities[filter], lower = 3150, upper = 3170,
          truth = TRUE,
          signal = TRUE,
          fitted = TRUE,
          postprocessed = TRUE,
          booktrue = as.matrix(toyspectrumsolution),
          cutoff.eps = 0.2)
```

`internals` *Internal functions*

Description

Functions not intended to be called directly be the user.

`modelfit-class` *"Modelfit"*

Description

Object returned from a call to fitModelParameters.

Slots

model The chosen model, either `"Gaussian"` or `"EMG"`.

fitting The mode of fitting which generated the object. Either `"most_intense"` or `"model"`.

alphafunction A function of m/z that computes an estimate for the `EMG` parameter `alpha`. See fitModelParameters for more detailed information on this parameter.

sigmafunction A function of m/z that computes an estimate for the `EMG` parameter `sigma`. See fitModelParameters for more detailed information on this parameter.

mufunction A function of m/z that computes an estimate for the `EMG` parameter `mu`. See fitModelParameters for more detailed information on this parameter.

peakfitresults A `matrix` of five columns if `model = "Gaussian"` and of six columns if `model = "EMG"`, respectively. The matrix contains basic information on the extracted peaks and the corresponding parameters. The first column contains the number of (mz, intensity)-pairs used for fitting, the second column contains the residual sums of squares of the model fit, the middle columns the parameter estimates and the last column the m/z position.

bestpeak A `list` containing detailed information on the 'best' peak, where 'best' is equivalent to 'minimum residual sums of squares' after fitting. Mainly used indirectly by calling the function `visualize`.

Methods

show Use `show(object)` for brief information about the object.

visualize A function to display graphically the result of parameter estimation. One can either visualize a single peak (`slot(object, "bestpeak")`) or the fit of the linear models postulated for the parameters `alpha`, `sigma` and `mu`. The function `visualize` is called with the following set of parameters.

`object` An object of class `modelfit`.

`type` A `character` specifying the object to be visualized. If `type = "peak"`, the fit of a single peak stored in `object@bestpeak` is displayed. If `type = "model"`, one obtains scatterplots of the form `parameter` vs. `mz` for each parameter in `parameters`, s. below.

`parameters` Needed if and only if `type = "model"` in order to choose the y-variable of the scatterplot. Several parameters may be specified at a time, in which case one obtains a multi-panel plot.

`modelfit` A `logical` indicating whether the estimated regression functions should be added to the scatterplots (if `type = "model"`).

---

myo500 *Myoglobine dataset*

Description

A protein mass spectrum (MALDI) of Myoglobine at resolution 500fmol.

Usage

```
data(myo500)
```

Source

The data have kindly be provided by B. Gregorius and A. Tholey, Department of Experimental Medicine, Working Group for Systematic Proteomics, Christian-Albrechts-Universitaet zu Kiel. The dataset may be used in publications as long as this source is quoted.

Examples

```
data(myo500)
plot(500)
```

peaklist-class *"Peaklist"*

Description

Object returned from a call to getPeaklist.

Slots

peaklist A matrix with rows equal to the number of peak patterns used in fitting and five columns. For each row (pattern), the first column contains the initial position of the peak pattern (loc_init, in Thomson), the second one contains the position of the most intense peak within a peak pattern (loc_most_intense, in Thomson), the third one contains the charge state (charge), the fourth one the quantification of the overall intensity of the pattern (quant) and the last one contains the amplitude at the most intense peak (amplitude).

peaklistprocessed The result after applying the postprocessing procedure to the raw list peaklist. It is a matrix of the same structure as peaklist, with two up to four additional columns: the fifth column contains the local noise level at the position, and the last one contains the ratio of the amplitude to local noise level (signal-to-noise ratio), hence quantitatively describing the significance of the peak pattern. If goodnessoffit has been set to TRUE, there are two additional columns labelled goodness_of_fit and ratio_adj. The column goodness_of_fit contains a local assessment of the goodness-of-fit at the respective positions in the spectrum, while the column ratio_adj contains a goodness-of-fit adjusted signal-to-noise ratio. If postprocessed has been set to FALSE when calling getPeaklist, this slot is an empty matrix.

model One of "Gaussian" or "EMG".

averagine.table Averagine table used, cf.getPeaklist.

loss One of "L2" or "L1".

alpha Function for the parameter "alpha" (model = "EMG").

sigma Function for the parameter "sigma" (both models).

mu Function for the parameter "mu" (model = "EMG").

charges A numeric vector containing the set of charge states used.

basis The matrix of basis functions if returnbasis has been set to TRUE when calling getPeaklist. Otherwise, this slot is an empty matrix.

book A matrix annotating each column of basis. Each row corresponds to a template/peak pattern characterized by the initial position of the peak pattern (column initial), the position of the most intense peak (column most_intense) and the charge state (column charge).

beta A numeric vector storing the amplitudes for each peak pattern in book/each column of basis, obtained by non-negative least squares estimation (if loss = "L2") or by non-negative least absolute deviation estimation (if loss = "L1"). Hence, together with book, cbind(book, beta) constitutes a predecessor of peaklist.

locnoise A matrix of local quantiles of the intensities computed along the spectrum. Each row corresponds to a m/z value of the spectrum (ordered increasingly), and each column to a quantile as indicated by the column names. For instance, slot(object, "locnoise")[,"0.5"] would give the vector of local medians.

noiselevel A numeric vector storing the noise level used for computation of the signal-to-noise ratios appearing in peaklistprocessed.

goodnessoffit A numeric vector storing the local goodness-of-fit criterion. An empty vector unless control.postprocessing$goodnessfittrue was TRUE when getPeaklist was called.

data A list storing the spectrum used to generate the object.

Methods

show Use `show(object)` for brief information about the object.

threshold Only applicable if `postprocessed` has been set to `TRUE` when calling `getPeaklist`. A function to threshold `peaklistprocessed`, yielding a peaklist maintaining only those peaks exceeding a specified threshold. Optionally, re-fitting of the spectrum using only those templates that have passed the thresholding step can be performed. The argument list is given by

`object` An object of class `peaklist`.

`threshold` The value to be used as threshold.

`ratio` Whether to use the signal-to-noise ratio `"ratio"` or the goodness-of-fit adjusted signal-to-noise ratio `"ratio_adj"`

`refit` Whether re-fitting of the spectrum should be done, using only the templates above the threshold.

`trace` A `logical` indicating whether information tracing the different steps of the fitting process should be displayed. Only used if `refit = TRUE`.

`eps` Function values below `eps` are set equal to precisely zero in order to make the basis function matrix sparse. Only used if `refit = TRUE`..

visualize A function to displaying the results of the fitting procedure, in particular significant peak patterns, in selected m/z ranges. The description of the arguments is as follows:

`object` An object of class `peaklist`.

`mz` The vector of m/z values used to obtain `object`.

`intensities` The vector of intensities used to obtain `object`.

`lower, upper` The m/z range for which the result of the fitting procedure should be visualized (`lower < upper`). *Hint*: the difference `upper - lower` should be between 0-30. If it is significantly larger, computation times increase considerably. Moreover, this will lead to a reduced quality of the plot.

`truth` May be set to `TRUE` if one has a precise knowledge of the true underlying signal (normally only occuring for simulated data) and one wants to compare the results with the gold standard. If `truth = TRUE`, `booktrue` (s. below) has to be specified. In this case, the true signal is displayed as upper panel in the plot.

`signal` A `logical` indicating whether the raw spectrum in the range `[lower,upper]` should be plotted. If `TRUE`, it is displayed as upper panel or as one of the middle panels.

`fitted` A `logical` indicating whether basically all (with restrictions, s. `cutoff.functions` and `cutoff.eps` below) fitted templates should be plotted. If `TRUE`, the result is displayed as a middle panel or as lower panel.

`postprocessed` A `logical` indicating whether fitted templates after postprocessing should be plotted. If `TRUE`, the result is displayed as lower panel.

`fittedfunction` A `logical` indicating whether the fitted function should be drawn as lines in the panel displaying the raw spectrum (requires `signal = TRUE`). Note that the fitted function is the result of summing up all fitted templates multiplied by their amplitude. Setting `fittedfunction = TRUE` requires that `object` contains the matrix of basis functions.

`fittedfunction.cut` A `logical` indicating whether the fitted function after cutting peak patterns falling below a prespecified cutoff should be drawn as lines in the panel displaying the raw spectrum (requires `signal = TRUE`). Setting `fittedfunction.cut = TRUE` requires that `object` contains the matrix of basis functions.

`quantile` Optional argument. If `quantile` is specified, it has to be one of the values `0.1,...,0.9`, and the corresponding local quantile as contained in the slot `locnoise` (s. above) will be added to the plot as dotted line.

booktrue Required if and only if truth = TRUE. In this case, booktrue has to be a matrix having the same structure as cbind(slot(object, "book"), slot(object, "beta")), i.e. the first two columns of booktrue contain initial- and most intense peak locations within a peak pattern, the third column contains the charge state and the fourth one the amplitude of the most intense peak. *Note:* the 'truth' is computed according to slot(object, model). This excludes the following scenario: assuming that the true peak shapes is "EMG", but the chosen model is "Gaussian". Then the true model in this function is computed according to the latter model.

cutoff.eps, cutoff.functions Control arguments reducing the number of templates to be plotted if fitted = TRUE. Specifying cutoff.eps removes all templates with intensity smaller than cutoff.eps. Specifying cutoff.functions as a positive integer only plots the cutoff 'most intense' templates.

... Additional options passed to plot.

Note that the colours in the plot have no meaning. They are only used to distinguish between different patterns within a panel. Further note that the colour of a pattern appearing in several panels may vary from panel to panel.

---

read.mzXML *Import of data in mzXML format*

Description

The function has been extracted from the package caMassClass. It allows the import of data in mzXML format typically used for LC-MS data.

Usage

```
read.mzXML(filename)
```

Arguments

filename	Path to the file that is to be imported.

Value

An object of class mzXML.

See Also

analyzeLCMS

sweepline *Sweep line scheme to process LC-MS data*

Description

The function implements a version of the sweep line scheme suggested in Schulz-Trieglaff et al. (2008) to aggregate the results of multiple scans of an LC-MS dataset.

Usage

```
sweepline(peaklists, rt, tol = 100, gap = 2, minboxlength = 5)
```

Arguments

peaklists	A list of matrices as in the slot peaklistprocessed of an object of class peaklist. It is assumed that the entries of peaklists correspond to the sequence retention times when ordered in ascending order.
rt	A numeric vector of retention times.
tol	A m/z tolerance (in parts per million, ppm) below which signals of different scans (retention times) are considered for merging.
gap	Retention time intervals are formed by merging signals of adjacent (with respect to retention time) scans. An intervals is closed (i.e. no further signals are added) once no additional signal has been added for more than gap consecutive scans.
minboxlength	Minimum number of adjacent signals found such that the corresponding retention time interval is kept in the output.

Value

A matrix having the following columns. Each row corresponds to a retention time interval of consecutive signals.

loc m/z-position corresponding to the interval.

charge charge of the signals found.

quant Cumulative intensities of the signals found.

rt_begin,rt_begin Boundaries of the retention time interval.

npeaks Total number of scans containing signals assigned to the interval.

gapcount Total number of scans corresponding to the retention time interval not containing a signal.

References

0. Schulz-Trieglaff and R. Hussong and C. Groepl and A. Leinenbach and A. Hildebrandt and C. Huber and K. Reinert. (2008) *Journal of Computational Biology, 15, 685-704*

See Also

analyzeLCMS

`toyspectrum` *Simulated protein mass spectrum*

Description

A simulated dataset for testing purposes. m/z ranges from 800 to about 4000 Thomson, sampled at approximately 118,000 points. The isotopic patterns present in this artificial dataset exactly obey the Averagine model in combination with Exponentially Modified Gaussians peak shapes, s. fitModelParameters if no noise were present. The noise has been generated from two sources: the first is a baseline pattern of Gaussian bumps with low intensity and the second one is additive noise drawn from a truncated Gaussian distribution. The dataset contains about 300, partially heavily overlapping, peak patterns whose data are stored in `data(toyspectrumsolution)`.

Usage

```
data(toyspectrum)
```

Examples

```
data(toyspectrum)
plot(toyspectrum)
```

Index