Package: chromatographR (via r-universe)

August 22, 2024

Type Package

Title Chromatographic Data Analysis Toolset

Version 0.7.1

Maintainer Ethan Bass <ethanbass@gmail.com>

Description Tools for high-throughput analysis of HPLC-DAD/UV chromatograms (or similar data). Includes functions for preprocessing, alignment, peak-finding and fitting, peak-table construction, data-visualization, etc. Preprocessing and peak-table construction follow the rough formula laid out in alsace (Wehrens, R., Bloemberg, T.G., and Eilers P.H.C., 2015. <doi:10.1093/bioinformatics/btv299>. Alignment of chromatograms is available using parametric time warping (ptw) (Wehrens, R., Bloemberg, T.G., and Eilers P.H.C. 2015. <doi:10.1093/bioinformatics/btv299>) or variable penalty dynamic time warping (VPdtw) (Clifford, D., & Stone, G. 2012. <doi:10.18637/jss.v047.i08>). Peak-finding uses the algorithm by Tom O'Haver <https://terpconnect.umd.edu/~toh/spectrum/PeakFindingandMeasurement.htm>. Peaks are then fitted to a gaussian or exponential-gaussian hybrid peak shape using non-linear least squares (Lan, K. & Jorgenson, J. W. 2001. <doi:10.1016/S0021-9673(01)00594-5>). See the vignette for more details and suggested workflow.

License GPL (>= 2)

URL https://ethanbass.github.io/chromatographR/, https://github.com/ethanbass/chromatographR/

BugReports https://github.com/ethanbass/chromatographR/issues/

Depends R (>= 3.5.0)

Imports caTools, chromConverter, dynamicTreeCut, fastcluster, Formula, fs, graphics, grDevices, lattice, methods, minpack.lm, parallel, ptw, purrr, scales, stats, utils, VPdtw

Suggests cowplot, ggplot2, knitr, openxlsx, pbapply, plotly, pvclust, reticulate, rmarkdown, rsvg, spelling, testthat (>= 3.0.0), vdiffr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.0

Repository https://ethanbass.r-universe.dev

RemoteUrl https://github.com/ethanbass/chromatographR

RemoteRef HEAD

RemoteSha b24f669d3c93bec0c82cf5b5a38ed9a745910769

Contents

chromatographR-package *chromatographR*

Description

Chromatographic Data Analysis Toolset

Details

Tools for high-throughput analysis of HPLC-DAD/UV chromatograms (or similar data). Includes functions for preprocessing, alignment, peak-finding and fitting, peak-table construction, data-visualization, etc. Preprocessing and peak-table construction follow the rough formula laid out in alsace (Wehrens, R., Bloemberg, T.G., and Eilers P.H.C., 2015. doi:10.1093/bioinformatics/btv299). Alignment of chromatograms is available using parametric time warping (ptw) (Wehrens, R., Bloemberg, T.G., and Eilers P.H.C. 2015. doi:10.1093/bioinformatics/btv299) or variable penalty dynamic time warping (VPdtw) (Clifford, D., & Stone, G. 2012. doi:10.18637/jss.v047.i08). Peak-finding relies on the algorithm suggested by Tom O'Haver in his Pragmatic Introduction to Signal Processing. Peaks are then fitted to a gaussian or exponential-gaussian hybrid peak shape using non-linear least squares (Lan, K. & Jorgenson, J. W. 2001. doi:10.1016/S00219673(01)005945). More details on package usage and a suggested workflow can be found in the vignette.

Author(s)

Ethan Bass

See Also

Useful links:

- `https://ethanbass.github.io/chromatographR/`
- `https://github.com/ethanbass/chromatographR/`
- Report bugs at `https://github.com/ethanbass/chromatographR/issues/`

attach_metadata *Attach experimental metadata*

Description

Attaches sample metadata to ‘peak_table‘ object. Metadata should be provided as a data.frame object. One of the columns in the supplied metadata must match exactly the row names of the peak table.

Usage

```
attach_metadata(peak_table, metadata, column)
```

Arguments

peak_table	A ‘peak_table‘ object.
metadata	A ‘data.frame‘ containing the sample metadata.
column	The name of the column in your metadata object containing the sample names. Sample names must match the row names of peak_table$tab.

Value

A peak_table object with attached metadata in the $sample_meta slot.

Author(s)

Ethan Bass

See Also

get_peaktable normalize_data

Examples

```
data(pk_tab)
path <- system.file("extdata", "Sa_metadata.csv", package = "chromatographR")
meta <- read.csv(path)
pk_tab <- attach_metadata(peak_table = pk_tab, metadata = meta, column="vial")
```

`attach_ref_spectra` *Attach reference spectra*

Description

Gathers reference spectra and attaches them to peak_table object. Reference spectra are defined either as the spectrum with the highest intensity (`max.int`) or as the spectrum with the highest average correlation to the other spectra in the peak_table (`max.cor`).

Usage

```
attach_ref_spectra(peak_table, chrom_list, ref = c("max.cor", "max.int"))
```

Arguments

`peak_table`	Peak table from `get_peaktable`.
`chrom_list`	A list of chromatograms in matrix format (timepoints x wavelengths). If no argument is provided here, the function will try to find the `chrom_list` object used to create the provided `peak_table`.
`ref`	What criterion to use to select reference spectra. Current options are maximum correlation (`max.cor`) or maximum signal intensity (`max.int`).

Value

A `peak_table` object with reference spectra attached in the `$ref_spectra` slot.

Author(s)

Ethan Bass

See Also

`get_peaks` `get_peaktable`

Examples

```
data(pk_tab)
pk_tab <- attach_ref_spectra(pk_tab, ref="max.int")
pk_tab <- attach_ref_spectra(pk_tab, ref = "max.cor")
```

boxplot.peak_table *Make boxplot from peak table.*

Description

The function can take multiple response variables on the left hand side of the formula (separated by +). In this case, a separate boxplot will be produced for each response variable.

Usage

```
## S3 method for class 'peak_table'
boxplot(x, formula, ...)
```

Arguments

x	A peak_table object
formula	A formula object
...	Additional arguments to boxplot

Examples

```
data(pk_tab)
path <- system.file("extdata", "Sa_metadata.csv", package = "chromatographR")
meta <- read.csv(path)
pk_tab <- attach_metadata(peak_table = pk_tab, metadata = meta, column="vial")
boxplot(pk_tab, formula=V11 ~ trt)
```

cluster_spectra *Cluster peaks by spectral similarity.*

Description

Function to cluster peaks by spectral similarity. A representative spectrum is selected for each peak in the provided peak table and used to construct a distance matrix based on spectral similarity (pearson correlation) between peaks. Hierarchical clustering with bootstrap resampling is performed on the resulting correlation matrix to classify peaks into by their spectral similarity.

Usage

```
cluster_spectra(
  peak_table,
  peak_no = c(5, 100),
  alpha = 0.95,
  nboot = 1000,
  plot_dend = TRUE,
  plot_spectra = TRUE,
  verbose = getOption("verbose"),
  save = FALSE,
  parallel = TRUE,
  max.only = FALSE,
  output = c("pvclust", "clusters"),
  ...
)
```

Arguments

peak_table	Peak table from get_peaktable.
peak_no	Minimum and maximum thresholds for the number of peaks a cluster may have.
alpha	Confidence threshold for inclusion of cluster.
nboot	Number of bootstrap replicates for pvclust.
plot_dend	Logical. If TRUE, plots dendrogram with bootstrap values.
plot_spectra	Logical. If TRUE, plots overlapping spectra for each cluster.
verbose	Logical. If TRUE, prints progress report to console.
save	Logical. If TRUE, saves pvclust object to current directory.
parallel	Logical. If TRUE, use parallel processing for pvclust.
max.only	Logical. If TRUE, returns only highest level for nested dendrograms.
output	What to return. Either clusters to return list of clusters, pvclust to return pvclust object, or both to return both items.
...	Additional arguments to pvclust.

Details

A representative spectrum is selected for each peak in the provided peak table and used to construct a distance matrix based on spectral similarity (pearson correlation) between peaks. It is suggested to attach representative spectra to the peak_table using attach_ref_spectra. Otherwise, representative spectra are obtained from the chromatogram with the highest absorbance at lambda max.

Hierarchical clustering with bootstrap resampling is performed on the resulting correlation matrix, as implemented in pvclust. Finally, bootstrap values can be used to select clusters that exceed a certain confidence threshold as defined by alpha. Clusters can also be filtered by the minimum and maximum size of the cluster using the argument peak_no. If max_only is TRUE, only the largest cluster in a nested dendrogram of clusters meeting the confidence threshold will be returned.

Value

Returns clusters and/or pvclust object according to the value of the output argument.

- If output = clusters, returns a list of S4 cluster objects.
- If output = pvclust, returns a pvclust object.
- If output = both, returns a nested list containing [[1]] the pvclust object, and [[2]] the list of S4 cluster objects.

The cluster objects consist of the following components:

- peaks: a character vector containing the names of all peaks contained in the given cluster.
- pval: a numeric vector of length 1 containing the bootstrap p-value (au) for the given cluster.

Note

- Users should be aware that the clustering algorithm will often return nested clusters. Thus, an individual peak could appear in more than one cluster.
- It is highly suggested to use more than 100 bootstraps if you run the clustering algorithm on real data even though we use nboot = 100 in the example to reduce runtime. The authors of pvclust suggest nboot = 10000.

Author(s)

Ethan Bass

References

R. Suzuki & H. Shimodaira. 2006. Pvclust: an R package for assessing the uncertainty in hierarchical clustering. *Bioinformatics*, **22(12)**:1540-1542. doi:10.1093/bioinformatics/btl117.

Examples

```
data(pk_tab)
data(Sa_warp)
pk_tab <- attach_ref_spectra(pk_tab, Sa_warp, ref="max.int")
cl <- cluster_spectra(pk_tab, nboot=100, max.only = FALSE, save = FALSE, alpha = .97)
```

combine_peaks *Combine peaks*

Description

Utility function to combine duplicate peaks in peak table, i.e. peaks that were integrated at more than one wavelength or component. Specify tolerance (tol) for retention time matching and minimum spectral correlation (min.cor) for a match.

Usage

```
combine_peaks(
  peak_table,
  tol = 0.01,
  min.cor = 0.9,
  choose = "max",
  verbose = getOption("verbose")
)
```

Arguments

`peak_table`	Peak table from `get_peaktable`.
`tol`	Tolerance for matching retention times (maximum retention time difference). Defaults to `.01`.
`min.cor`	Minimum spectral correlation to confirm a match. Defaults to `0.9`.
`choose`	If `max` will retain peak with highest intensity. Otherwise, the first column in the data.frame will be retained.
`verbose`	Logical. Whether to print status to the console.

Value

A peak table similar to the input peak table, but with duplicate columns combined according to the specified criteria.

Author(s)

Ethan Bass

See Also

`get_peaks`

Examples

```
data(pk_tab)
data(Sa_warp)
pk_tab <- attach_ref_spectra(pk_tab)
combine_peaks(pk_tab, tol = .02, min.cor = .9)
```

`correct_peaks` *Correct peak positions according to a ptw warping model*

Description

Corrects retention time differences using parametric time warping as implemented in `ptw`.

Usage

```
correct_peaks(peak_list, mod_list, chrom_list, match_names = TRUE)
```

Arguments

`peak_list`	A ‘peak_list‘ object created by `get_peaks`, containing a nested list of peak tables where the first level is the sample, and the second level is the spectral wavelength. Every component is described by a matrix where every row is one peak, and the columns contain information on retention time, peak width (FWHM), peak width, height, and area.
`mod_list`	A list of ptw models.
`chrom_list`	List of chromatograms supplied to create ptw models.
`match_names`	Logical. Whether to actively match the names of the `peak_list` to the list of models (`mod_list`). Defaults to `TRUE`.

Details

Once an appropriate warping model has been established, corrected retention times can be predicted for each peak. These are stored in a separate column in the list of peak tables.

Value

The input list of peak tables is returned with extra columns containing the corrected retention time.

Note

This function is adapted from getPeakTable function in the alsace package by Ron Wehrens.

Author(s)

Ron Wehrens, Ethan Bass

See Also

`correct_rt`

`correct_rt` *Correct retention time*

Description

Aligns chromatograms using one of two algorithms, according to the value of `alg`: either parametric time warping, as implemented in `ptw`, or variable penalty dynamic time warping, as implemented in `VPdtw`. The `init.coef` and `n.traces` arguments apply only to `ptw` warping, while `penalty` and `maxshift` apply only to `vpdtw` warping.

Usage

```
correct_rt(
  chrom_list,
  lambdas,
  models = NULL,
  reference = "best",
  alg = c("ptw", "vpdtw"),
  what = c("corrected.values", "models"),
  init.coef = c(0, 1, 0),
  n.traces = NULL,
  n.zeros = 0,
  scale = FALSE,
  trwdth = 200,
  plot_it = FALSE,
  penalty = 5,
  maxshift = 50,
  verbose = getOption("verbose"),
  show_progress = NULL,
  cl = 2,
  ...
)
```

Arguments

chrom_list	List of chromatograms in matrix format.
lambdas	Select wavelengths to use by name.
models	List of models to warp by. The models provided here (if any) must match the algorithm selected in alg.
reference	Index of the sample that is to be considered the reference sample.
alg	algorithm to use: parametric time warping (ptw) or variable penalty dynamic time warping (vpdtw).
what	What to return: either the 'corrected.values' (useful for visual inspection) or the warping 'models' (for further programmatic use).
init.coef	Starting values for the optimization.
n.traces	Number of traces to use.
n.zeros	Number of zeros to add.
scale	Logical. If true, scale chromatograms before warping.
trwdth	width of the triangle in the WCC criterion.
plot_it	Logical. Whether to plot alignment.
penalty	The divisor used to calculate the penalty for VPdtw. The warping penalty is calculated by dividing the dilation by this number. Thus, a higher number will produce a lower penalty and be more permissive, while a lower number will produce a higher penalty and allow less warping. Defaults to 5.
maxshift	Integer. Maximum allowable shift for VPdtw. Defaults to 50.

`verbose` Whether to print verbose output.

`show_progress` Logical. Whether to show progress bar. Defaults to `TRUE` if `pbapply` is installed. Currently works only for `ptw` alignments.

`cl` Argument to `pblapply` or `mclapply`. Either an integer specifying the number of clusters to use for parallel processing or a cluster object created by `makeCluster`. Defaults to 2. On Windows integer values will be ignored.

`...` Optional arguments for the `ptw` function. The only argument that cannot be changed is `warp.type`: this is always equal to `"global"`.

Value

A list of warping models or a list of warped absorbance profiles, depending on the value of the `what` argument.

Note

Adapted from correctRT function in the alsace package by Ron Wehrens.

Author(s)

Ethan Bass

References

- Clifford, D., Stone, G., Montoliu, I., Rezzi, S., Martin, F. P., Guy, P., Bruce, S., & Kochhar, S. 2009. Alignment using variable penalty dynamic time warping. *Analytical chemistry*, **81(3)**:1000-1007. doi:10.1021/ac802041e.
- Clifford, D., & Stone, G. 2012. Variable Penalty Dynamic Time Warping Code for Aligning Mass Spectrometry Chromatograms in R. *Journal of Statistical Software*, **47(8)**:1-17. doi:10.18637/jss.v047.i08.
- Eilers, P.H.C. 2004. Parametric Time Warping. *Anal. Chem.*, **76**:404-411. doi:10.1021/ac034800e.
- Wehrens, R., Bloemberg, T.G., and Eilers P.H.C. 2015. Fast parametric time warping of peak lists. *Bioinformatics*, **31**:3063-3065. doi:10.1093/bioinformatics/btv299.
- Wehrens, R., Carvalho, E., Fraser, P.D. 2015. Metabolite profiling in LC–DAD using multivariate curve resolution: the alsace package for R. *Metabolomics*, **11**:143-154. doi:10.1007/s1130601406835.

See Also

`ptw`, `correct_peaks`, `VPdtw`

Examples

```
data(Sa_pr)
warping.models <- correct_rt(Sa_pr, what = "models", lambdas=c(210))
warp <- correct_rt(chrom_list = Sa_pr, models = warping.models)
```

filter_peaks *Filter peak lists*

Description

Utility function to remove peaks from a peak list, e.g. because their intensity is too low. Currently one can filter on peak height, peak area, standard deviation, and/or retention time.

Usage

```
filter_peaks(peak_list, min_height, min_area, min_sd, max_sd, min_rt, max_rt)
```

Arguments

`peak_list`	A peak_list object, consisting of a nested list of peak tables, where the first level is the sample, and the second level is the spectral component. Every component is described by a matrix where every row is one peak, and the columns contain information on retention time, full width at half maximum (FWHM), peak width, height, and area.
`min_height`	Minimum peak height.
`min_area`	Minimum peak area.
`min_sd`	Minimal standard deviation.
`max_sd`	Maximum standard deviation.
`min_rt`	Minimum retention time.
`max_rt`	Maximum retention time.

Value

A peak list similar to the input, with all rows removed from that do not satisfy the specified criteria.

Author(s)

Ron Wehrens, Ethan Bass

See Also

`get_peaks`, `filter_peaktable`

filter_peaktable *Filter peak table*

Description

Utility function to remove peaks from peak table, e.g. because their intensity is too low. Currently one can filter on mean or median peak intensity, or retention time.

Usage

```
filter_peaktable(
  peak_table,
  rts,
  min_rt,
  max_rt,
  min_value,
  lambda,
  what = c("median", "mean", "max"),
  tol = 0
)
```

Arguments

peak_table	A peak_table object from get_peaktable.
rts	Vector of retention times to include in the peak table.
min_rt	Minimum retention time to include in the peak table.
max_rt	Maximum retention time to include in the peak table.
min_value	Minimal cutoff for summarized peak intensity.
lambda	Component(s) to include in peak table (e.g. wavelengths if you are using HPLC-DAD/UV).
what	Whether to summarize intensities using mean, median, or max.
tol	Tolerance for matching of retention times to rts.

Value

A peak table similar to the input, with all columns removed from the peak table that do not satisfy the specified criteria.

Author(s)

Ethan Bass

See Also

get_peaktable, filter_peaks

Examples

```
data(pk_tab)
pk_tab <- filter_peaktable(pk_tab, min_rt = 10, max_rt = 16)
```

`find_peaks` *Find peaks in chromatographic profile*

Description

Find peaks in chromatographic profile.

Usage

```
find_peaks(
  y,
  smooth_type = c("gaussian", "box", "savgol", "mva", "tmva", "none"),
  smooth_window = 0.001,
  slope_thresh = 0,
  amp_thresh = 0,
  bounds = TRUE
)
```

Arguments

`y`	Signal (as a numerical vector).
`smooth_type`	Type of smoothing. Either gaussian kernel (`"gaussian"`), box kernel (`"box"`), savitzky-golay smoothing (`"savgol"`), moving average (`"mva"`), triangular moving average (`"tmva"`), or no smoothing (`"none"`).
`smooth_window`	Smoothing window. Larger values of this parameter will exclude sharp, narrow features. If the supplied value is between 0 and 1, window will be interpreted as a proportion of points to include. Otherwise, the window will be the absolute number of points to include in the window. (Defaults to `.001`).
`slope_thresh`	Minimum threshold for slope of the smoothed first derivative. This parameter filters on the basis of peak width, such that larger values will exclude broad peaks from the peak list. (Defaults to `0`).
`amp_thresh`	Minimum threshold for peak amplitude. This parameter filters on the basis of peak height, such that larger values will exclude small peaks from the peak list. (Defaults to `0`).
`bounds`	Logical. If TRUE, includes peak boundaries in data.frame. (Defaults to `TRUE`).

Details

Find peaks by looking for zero-crossings in the smoothed first derivative of the signal (`y`) that exceed the specified slope threshold (`slope_thresh`). Additionally, peaks can be filtered by supplying a minimal amplitude threshold (`amp_thresh`), filtering out peaks below the specified height. Smoothing is intended to prevent the algorithm from getting caught up on local minima and maxima that do not represent true features. Several smoothing options are available, including `"gaussian"`, box kernel (`"box"`), savitzky-golay smoothing (`"savgol"`), moving average (`"mva"`), triangular moving average (`"tmva"`), or no smoothing (`"none"`).

It is recommended to do pre-processing using the `preprocess` function before peak detection. Overly high chromatographic resolution can sometimes cause peaks to be split into multiple segments. In this case, it is recommended to increase the `smooth_window` or reduce the resolution along the time axis by adjusting the `dim1` argument during preprocessing.

Value

If bounds == `TRUE`, returns a data.frame containing the center, start, and end of each identified peak. Otherwise, returns a numeric vector of peak centers. All locations are expressed as indices.

Note

The `find_peaks` function is adapted from MATLAB code included in Prof. Tom O'Haver's Pragmatic Introduction to Signal Processing.

Author(s)

Ethan Bass

References

O'Haver, Tom. Pragmatic Introduction to Signal Processing: Applications in scientific measurement. `https://terpconnect.umd.edu/~toh/spectrum/` (Accessed January, 2022).

See Also

`fit_peaks`, `get_peaks`

Examples

```
data(Sa_pr)
find_peaks(Sa_pr[[1]][,"220"])
```

`fit_peaks` *Fit chromatographic peaks to an exponential-gaussian hybrid or gaussian profile*

Description

Fit peak parameters using exponential-gaussian hybrid or gaussian function.

Usage

```
fit_peaks(
  x,
  lambda,
  pos = NULL,
  sd.max = 50,
  fit = c("egh", "gaussian", "raw"),
  max.iter = 1000,
  estimate_purity = TRUE,
  noise_threshold = 0.001,
  ...
)
```

Arguments

x	A chromatogram in matrix format.
lambda	Wavelength to fit peaks at.
pos	Locations of peaks in vector y. If NULL, find_peaks will run automatically to find peak positions.
sd.max	Maximum width (standard deviation) for peaks. Defaults to 50.
fit	Function for peak fitting. (Currently exponential-gaussian hybrid egh, gaussian and raw settings are supported). If raw is selected, trapezoidal integration will be performed on raw data without fitting a peak shape. Defaults to egh.)
max.iter	Maximum number of iterations to use in nonlinear least squares peak-fitting. (Defaults to 1000).
estimate_purity	Logical. Whether to estimate purity or not. Defaults to TRUE.
noise_threshold	Noise threshold. Input to get_purity.
...	Additional arguments to find_peaks.

Details

Peak parameters are calculated by fitting the data to a gaussian or exponential-gaussian hybrid curve using non-linear least squares estimation as implemented in nlsLM. Area under the fitted curve is then estimated using trapezoidal approximation.

Value

The `fit_peaks` function returns a matrix, whose columns contain the following information about each peak:

`rt`	Location of the peak maximum.
`start`	Start of peak (only included in table if `bounds = TRUE`).
`end`	End of peak (only included in table if `bounds = TRUE`).
`sd`	The standard deviation of the peak.
`tau`	τ parameter (only included in table if `fit = "egh"`).
`FWHM`	The full width at half maximum.
`height`	Peak height.
`area`	Peak area.
`r.squared`	The R-squared value for linear fit of the model to the data.
`purity`	The spectral purity of peak as assessed by `get_purity`.

Again, the first five elements (rt, start, end, sd and FWHM) are expressed as indices, so not in terms of the real retention times. The transformation to "real" time is done in function `get_peaks`.

Note

The `fit_peaks` function is adapted from Dr. Robert Morrison's DuffyTools package as well as code published in Ron Wehrens' alsace package.

Author(s)

Ethan Bass

References

* Lan, K. & Jorgenson, J. W. 2001. A hybrid of exponential and gaussian functions as a simple model of asymmetric chromatographic peaks. *Journal of Chromatography A* **915**:1-13. doi:10.1016/S00219673(01)005945.

* Naish, P. J. & Hartwell, S. 1988. Exponentially Modified Gaussian functions - A good model for chromatographic peaks in isocratic HPLC? *Chromatographia*, /bold26: 285-296. doi:10.1007/BF02268168.

See Also

`find_peaks`, `get_peaks`

Examples

```
data(Sa_pr)
fit_peaks(Sa_pr[[1]], lambda = 220)
```

get_peaks *Get peak list.*

Description

Finds and fits peaks and extracts peak parameters from a list of chromatograms at the specified wavelengths.

Usage

```
get_peaks(
  chrom_list,
  lambdas,
  fit = c("egh", "gaussian", "raw"),
  sd.max = 50,
  max.iter = 100,
  time.units = c("min", "s", "ms"),
  estimate_purity = FALSE,
  noise_threshold = 0.001,
  show_progress = NULL,
  cl = 2,
  collapse = FALSE,
  ...
)
```

Arguments

chrom_list A list of profile matrices, each of the same dimensions (timepoints × wavelengths).

lambdas Character vector of wavelengths to find peaks at.

fit What type of fit to use. Current options are exponential-gaussian hybrid (egh), gaussian or raw. The raw setting performs trapezoidal integration directly on the raw data without fitting a peak shape.

sd.max Maximum width (standard deviation) for peaks. Defaults to 50.

max.iter Maximum number of iterations for non-linear least squares in fit_peaks.

time.units Units of sd, FWHM, area, and tau (if applicable). Options are minutes ("min"), seconds ("s"), or milliseconds ("ms").

estimate_purity
Logical. Whether to estimate purity or not. Defaults to FALSE. (If TRUE, this will slow down the function significantly).

noise_threshold
Noise threshold. Argument to get_purity.

show_progress Logical. Whether to show progress bar. Defaults to TRUE if pbapply is installed.

`cl` Argument to `pblapply` or `mclapply`. Either an integer specifying the number of clusters to use for parallel processing or a cluster object created by `makeCluster`. Defaults to 2. On Windows integer values will be ignored.

`collapse` Logical. Whether to collapse multiple peak lists per sample into a single list when multiple wavelengths (`lambdas`) are provided.

`...` Additional arguments to `find_peaks`. Arguments provided to `find_peaks` can be used to fine-tune the peak-finding algorithm. Most importantly, the `smooth_window` should be increased if features are being split into multiple bins. Other arguments that can be used here include `smooth_type`, `slope_thresh`, and `amp_thresh`.

Details

Peaks are located by finding zero-crossings in the smoothed first derivative of the specified chromatographic traces (function `find_peaks`). At the given positions, an exponential-gaussian hybrid (or regular gaussian) function is fit to the signal using `fit_peaks` according to the value of `fit`. Finally, the area is calculated using trapezoidal approximation.

Additional arguments can be provided to `find_peaks` to fine-tune the peak-finding algorithm. For example, the `smooth_window` can be increased to prevent peaks from being split into multiple features. Overly aggressive smoothing may cause small peaks to be overlooked.

The standard deviation (`sd`), full-width at half maximum (`FWHM`), tau `tau`, and `area` are returned in units determined by `time.units`. By default, the units are in minutes. To compare directly with 'ChemStation' integration results, the time units should be in seconds.

Value

The result is an S3 object of class `peak_list`, containing a nested list of data.frames containing information about the peaks fitted for each chromatogram at each of wavelengths specified by the `lamdas` argument. Each row in these data.frames is a peak and the columns contain information about various peak parameters:

- `rt`: The retention time of the peak maximum.
- `start`: The retention time where the peak is estimated to begin.
- `end`: The retention time where the peak is estimated to end.
- `sd`: The standard deviation of the fitted peak shape.
- `tau` The value of parameter τ. This parameter determines peak asymmetry for peaks fit with an exponential-gaussian hybrid function. (This column will only appear if `fit = egh`.
- `FWHM`: The full-width at half maximum.
- `height`: The height of the peak.
- `area`: The area of the peak as determined by trapezoidal approximation.
- `r.squared` The coefficient of determination (R^2) of the fitted model to the raw data. (**Note**: this value is calculated by fitting a linear model of the fitted peak values to the raw data. This approach is statistically questionable, since the models are fit using non-linear least squares. Nevertheless, it can still be useful as a rough metric for "goodness-of-fit").
- `purity` The peak purity.

Note

The bones of this function are adapted from the getAllPeaks function authored by Ron Wehrens (though the underlying algorithms for peak identification and peak-fitting are not the same).

Author(s)

Ethan Bass

References

- Lan, K. & Jorgenson, J. W. 2001. A hybrid of exponential and gaussian functions as a simple model of asymmetric chromatographic peaks. *Journal of Chromatography A* **915**:1-13. doi:10.1016/S00219673(01)005945.
- Naish, P. J. & Hartwell, S. 1988. Exponentially Modified Gaussian functions - A good model for chromatographic peaks in isocratic HPLC? *Chromatographia*, **26**: 285-296. doi:10.1007/BF02268168.
- O'Haver, Tom. Pragmatic Introduction to Signal Processing: Applications in scientific measurement. `https://terpconnect.umd.edu/~toh/spectrum/` (Accessed January, 2022).
- Wehrens, R., Carvalho, E., Fraser, P.D. 2015. Metabolite profiling in LC–DAD using multivariate curve resolution: the alsace package for R. *Metabolomics* **11**:143-154. doi:10.1007/s1130601406835.

See Also

`find_peaks`, `fit_peaks`

Examples

```
data(Sa_pr)
pks <- get_peaks(Sa_pr, lambdas = c('210'), sd.max=50, fit="egh")
```

---

get_peaktable *Convert peak list into an ordered peak table.*

Description

Returns a peak_table object. The first slot contains a matrix of intensities, where rows correspond to samples and columns correspond to aligned features. The rest of the slots contain various meta-data about peaks, samples, and experimental settings.

Usage

```
get_peaktable(
  peak_list,
  chrom_list,
  response = c("area", "height"),
  use.cor = NULL,
  hmax = 0.2,
  plot_it = FALSE,
  ask = plot_it,
  clust = c("rt", "sp.rt"),
  sigma.t = NULL,
  sigma.r = 0.5,
  deepSplit = FALSE,
  verbose = FALSE,
  out = c("data.frame", "matrix")
)
```

Arguments

`peak_list` A `peak_list` object created by `get_peaks`, containing a nested list of peak tables: the first level is the sample, and the second level is the spectral wavelength. Every component is described by a data.frame where every row is one peak, and the columns contain information on various peak parameters.

`chrom_list` A list of chromatographic matrices.

`response` Indicates whether peak area or peak height is to be used as intensity measure. Defaults to `area` setting.

`use.cor` Logical. Indicates whether to use corrected retention times (`rt.cor` column) or raw retention times (`rt` column). Unless otherwise specified, the `rt.cor` column will be used by default if it exists in the provided `peak_list`.

`hmax` Height at which the complete linkage dendrogram will be cut. Can be interpreted as the maximal inter-cluster retention time difference.

`plot_it` Logical. If TRUE, for every component a stripplot will be shown indicating the clustering.

`ask` Logical. Ask before showing new plot? Defaults to TRUE.

`clust` Specify whether to perform hierarchical clustering based on spectral similarity and retention time (`sp.rt`) or retention time alone (`rt`). Defaults to `rt`. The `sp.rt` option is experimental and should be used with caution.

`sigma.t` Width of gaussian in retention time distance function. Controls weight given to retention time if `sp.rt` is selected.

`sigma.r` Width of gaussian in spectral similarity function. Controls weight given to spectral correlation if `sp.rt` is selected.

`deepSplit` Logical. Controls sensitivity to cluster splitting. If `TRUE`, function will return more smaller clusters. See documentation for `cutreeDynamic` for additional information.

`verbose` Logical. Whether to print warning when combining peaks into single time window. Defaults to `FALSE`.

`out` Specify `data.frame` or `matrix` as output. Defaults to `data.frame`.

Details

The function performs a complete linkage clustering of retention times across all samples, and cuts at a height given by the user (which can be understood as the maximal inter-cluster retention time difference) in the simple case based on retention times. Clustering can also incorporate information about spectral similarity using a distance function adapted from Broeckling et al., 2014:

$$latexascii$$

If two peaks from the same sample are assigned to the same cluster, a warning message is printed to the console. These warnings can usually be ignored, but one could also consider reducing the `hmax` variable. However, this may lead to splitting of peaks across multiple clusters. Another option is to filter the peaks by intensity to remove small features.

Value

The function returns an S3 `peak_table` object, containing the following elements:

- `tab`: the peak table itself – a data-frame of intensities in a sample x peak configuration.
- `pk_meta`: A data.frame containing peak meta-data (e.g. the spectral component, peak number, and average retention time).
- `sample_meta`: A data.frame of sample meta-data. Must be added using `attach_metadata`).
- `ref_spectra`: A data.frame of reference spectra (in a wavelength x peak configuration). Must be added using `attach_ref_spectra`
- `args`: A vector of arguments given to `get_peaktable` to generate the peak table.

Note

This function is adapted from getPeakTable function in the alsace package by Ron Wehrens.

Author(s)

Ethan Bass

References

- Broeckling, C. D., Afsar F.A., Neumann S., Ben-Hur A., and Prenni J.E. 2014. RAMClust: A Novel Feature Clustering Method Enables Spectral-Matching-Based Annotation for Metabolomics Data. *Anal. Chem.* **86**:6812-6817. doi:10.1021/ac501530d.
- Wehrens, R., Carvalho, E., Fraser, P.D. 2015. Metabolite profiling in LC–DAD using multivariate curve resolution: the alsace package for R. *Metabolomics* **11**:143-154. doi:10.1007/s1130601406835.

See Also

`attach_ref_spectra` `attach_metadata`

Examples

```
data(Sa_pr)
pks <- get_peaks(Sa_pr, lambdas = c('210'))
get_peaktable(pks, response = "area")
```

---

merge_peaks *Merge split peaks*

Description

Utility function to combine split peaks into a single column of the peak table.

Usage

```
merge_peaks(peak_table, peaks, method = c("max", "sum"))
```

Arguments

peak_table	Peak table from get_peaktable.
peaks	A vector specifying the names or indices of peaks to be merged.
method	Method to merge peaks. Either max to select the largest peak from each sample or sum to sum the peaks together.

Details

Merges the specified peaks in peak table, by selecting the largest value from each column if method is "max". If method is "sum", merges peak by summing their values.

Value

A peak table similar to the input peak table, but where the specified columns are combined.

Author(s)

Ethan Bass

Examples

```
data(pk_tab)
pk_tab <- merge_peaks(peak_table = pk_tab, peaks=c("V10","V11"))
```

mirror_plot *Make mirror plot from peak table.*

Description

Plots chromatograms as a mirror plot.

Usage

```
mirror_plot(
  x,
  chrom_list,
  lambdas = NULL,
  var,
  subset = NULL,
  print_legend = TRUE,
  legend_txt = NULL,
  legend_pos = "topright",
  legend_size = 1,
  mirror = TRUE,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`x`	The peak table (output from `get_peaktable` function).
`chrom_list`	A list of chromatograms in matrix format (timepoints x wavelengths). If no argument is provided here, the function will try to find the `chrom_list` object used to create the `peak_table`.
`lambdas`	The wavelength you wish to plot the traces at.
`var`	Variable to index chromatograms.
`subset`	Character vector specifying levels to use (if more than 2 levels are present in `var`).
`print_legend`	Logical. Whether to print legend. Defaults to `TRUE`.
`legend_txt`	Character vector containing labels for legend.
`legend_pos`	Legend position.
`legend_size`	Legend size (`cex` argument). Default is 1.
`mirror`	Logical. Whether to plot as mirror or stacked plots. Defaults to `TRUE`.
`xlim`	Numerical vector specifying limits for x axis.
`ylim`	Numerical vector specifying limits for y axis.
`...`	Additional arguments to `matplot` function.

Details

Can be used to confirm the identity of a peak or check that a particular column in the peak table represents a single compound. Can also be used to create simple box-plots to examine the distribution of a peak with respect to variables defined in sample metadata.

Value

No return value, called for side effects.

Side effects

If `mirror_plot` is TRUE, plots a mirror plot comparing two treatments defined by `var` and `subset` (if more than two factors are present in `var`). Otherwise, if `mirror_plot` is FALSE, the treatments are plotted in two separate panes.

Author(s)

Ethan Bass

Examples

```
data(Sa_warp)
data(pk_tab)
path <- system.file("extdata", "Sa_metadata.csv", package = "chromatographR")
meta <- read.csv(path)
pk_tab <- attach_metadata(peak_table = pk_tab, metadata = meta, column="vial")
mirror_plot(pk_tab,lambdas = c("210","260"), var = "trt", mirror = TRUE,
  col = c("green","blue"))
```

`normalize_data` *Normalize peak table or chromatograms*

Description

Normalizes peak table or list of chromatograms by specified column in sample metadata. Metadata must first be attached to `peak_table` using `attach_metadata`.

Usage

```
normalize_data(
  peak_table,
  column,
  chrom_list,
  what = c("peak_table", "chrom_list"),
  by = c("meta", "peak")
)
```

Arguments

`peak_table` A ‘peak_table‘ object

`column` The name of the column containing the weights.

`chrom_list` List of chromatograms for normalization. The samples must be in same order as the peak_table. If no argument is provided here, the function will try to find the `chrom_list` object used to create the provided `peak_table`.

`what` ‘peak_table‘ or list of chromatograms (‘chrom_list‘).

`by` Whether to normalize by a column in sample metadata (`meta`) or by a column in the peak table itself (`peak`).

Value

A `peak_table` object where the peaks are normalized by the mass of each sample.

Author(s)

Ethan Bass

See Also

`get_peaktable` `attach_metadata`

Examples

```
data(pk_tab)
path <- system.file("extdata", "Sa_metadata.csv", package = "chromatographR")
meta <- read.csv(path)
pk_tab <- attach_metadata(peak_table = pk_tab, metadata = meta, column="vial")
norm <- normalize_data(pk_tab, "mass", what = "peak_table")
```

`pk_tab` *Goldenrod peak table*

Description

Peak table generated from exemplary goldenrod root extracts.

Usage

```
data(pk_tab)
```

Format

A `peak_table` object.

plot.peak_list *Plot fitted peak shapes.*

Description

Visually assess integration accuracy by plotting fitted peaks over trace.

Usage

```
## S3 method for class 'peak_list'
plot(
  x,
  ...,
  chrom_list,
  idx = 1,
  lambda = NULL,
  points = FALSE,
  ticks = FALSE,
  a = 0.5,
  color = NULL,
  cex.points = 0.5,
  numbers = FALSE,
  cex.font = 0.5,
  y.offset = 25,
  plot_purity = FALSE,
  res,
  index = NULL
)
```

Arguments

x	A peak_list object. Output from the get_peaks function.
...	Additional arguments to main plot function.
chrom_list	List of chromatograms (retention time x wavelength matrices)
idx	Index or name of chromatogram to be plotted.
lambda	Wavelength for plotting.
points	Logical. If TRUE, plot peak maxima. Defaults to FALSE.
ticks	Logical. If TRUE, mark beginning and end of each peak. Defaults to FALSE.
a	Alpha parameter controlling the transparency of fitted shapes.
color	The color of the fitted shapes.
cex.points	Size of points. Defaults to 0.5
numbers	Whether to number peaks. Defaults to FALSE.
cex.font	Font size if peaks are numbered. Defaults to 0.5.

`y.offset`	Y offset for peak numbers. Defaults to 25.
`plot_purity`	Whether to add visualization of peak purity.
`res`	time resolution for peak fitting
`index`	This argument is deprecated. Please use `idx` instead.

Value

No return value, called for side effects.

Side effects

Plots a chromatographic trace from the specified chromatogram (`chr`) at the specified wavelength (`lambda`) with fitted peak shapes from the provided `peak_list` drawn underneath the curve.

Author(s)

Ethan Bass

See Also

`get_peaks`

`plot.peak_table` *Plot spectrum from peak table*

Description

Plots the trace and/or spectrum for a given peak in peak table.

Usage

```
## S3 method for class 'peak_table'
plot(
  x,
  loc,
  chrom_list,
  what = "peak",
  idx = "max",
  lambda = "max",
  plot_spectrum = TRUE,
  plot_trace = TRUE,
  box_plot = FALSE,
  vars = NULL,
  spectrum_labels = TRUE,
  scale_spectrum = FALSE,
  export_spectrum = FALSE,
  verbose = TRUE,
  engine = c("base", "plotly", "ggplot"),
  chr = NULL,
  ...
)
```

Arguments

`x` The peak table (output from `get_peaktable` function).

`loc` A vector specifying the peak(s) or retention time(s) that you wish to plot.

`chrom_list` A list of chromatograms in matrix format (timepoints x wavelengths). If no argument is provided here, the function will try to find the `chrom_list` object used to create the `peak_table`.

`what` What to look for. Either `peak` to extract spectral information for a certain peak, `rt` to scan by retention time, or `click` to manually select retention time by clicking on the chromatogram. Defaults to `peak`.

`idx` Numerical index of chromatogram you wish to plot; "max" to plot the chromatogram with the largest signal; or "all" to plot spectra for all chromatograms.

`lambda` The wavelength you wish to plot the trace at (if `plot_chrom` is TRUE and/or the wavelength to be used for the determination of signal abundance.

`plot_spectrum` Logical. If TRUE, plots the spectrum of the chosen peak. Defaults to TRUE.

`plot_trace` Logical. If TRUE, plots the trace of the chosen peak at lambda. Defaults to TRUE.

`box_plot` Logical. If TRUE, plots box plot using categories defined by `vars`.

`vars` Independent variables for boxplot. Righthand side of formula.

`spectrum_labels`
Logical. If TRUE, plots labels on maxima in spectral plot. Defaults to TRUE.

`scale_spectrum` Logical. If TRUE, scales spectrum to unit height. Defaults to FALSE.

`export_spectrum`
Logical. If TRUE, exports spectrum to console. Defaults to FALSE.

`verbose` Logical. If TRUE, prints verbose output to console. Defaults to TRUE.

`engine` Which plotting engine to use: either `base` or `plotly`.

`chr` Deprecated. Please use `idx` instead.

`...` Additional arguments to `boxplot`.

Details

Can be used to confirm the identity of a peak or check that a particular column in the peak table represents a single compound. Can also be used to create simple box-plots to examine the distribution of a peak with respect to variables defined in sample metadata.

Value

If `export_spectrum` is TRUE, returns the spectrum as a `data.frame` with wavelengths as rows and columns encoding the absorbance (or normalized absorbance, if `scale_spectrum` is TRUE) for the specified sample(s). Otherwise, there is no return value.

Side effects

If plot_trace is TRUE, plots the chromatographic trace of the specified chromatogram (chr), at the specified wavelength (lambda) with a dotted red line to indicate the retention time given by loc. The trace is a single column from the chromatographic matrix.

If plot_spectrum is TRUE, plots the spectrum for the specified chromatogram at the specified retention time. The spectrum is a single row from the chromatographic matrix.

If box_plot is TRUE, produces a boxplot from the specified peak with groups provided by vars.

Author(s)

Ethan Bass

plot.ptw_list *Plot PTW alignments*

Description

Plot PTW alignments

Usage

```
## S3 method for class 'ptw_list'
plot(x, lambdas, legend = TRUE, ...)
```

Arguments

x	A ptw_list object created by correct_rt.
lambdas	Which lambdas to plot.
legend	Logical. Whether to label the plots.
...	Additional arguments to matplot.

Author(s)

Ethan Bass

plot_all_spectra *Plot all spectra for chosen peak.*

Description

Plot multiple for a given peak in peak table. Wrapper for plot_spectrum.

Usage

```
plot_all_spectra(
  peak,
  peak_table,
  chrom_list,
  idx = "all",
  chrs = NULL,
  engine = c("base", "ggplot2", "plotly"),
  plot_spectrum = TRUE,
  export_spectrum = TRUE,
  scale_spectrum = TRUE,
  overlapping = TRUE,
  verbose = FALSE,
  what = c("peak", "rt", "idx"),
  ...
)
```

Arguments

`peak` The name of a peak to plot (in character format)

`peak_table` The peak table (output from get_peaktable function)

`chrom_list` A list of chromatograms in matrix format (timepoints x components). If no argument is provided here, the function will try to find the `chrom_list` object used to create the provided `peak_table`.

`idx` Vector of chromatograms to plot.

`chrs` Deprecated. Please use `idx` instead.

`engine` Which plotting engine to use: `base`, `ggplot2`, or `plotly`.

`plot_spectrum` Logical. If TRUE, plots the spectrum of the chosen peak.

`export_spectrum` Logical. If TRUE, exports spectrum to console. Defaults to FALSE.

`scale_spectrum` Logical. If TRUE, scales spectrum to unit height.

`overlapping` Logical. If TRUE, plot spectra in single plot.

`verbose` Logical. If TRUE, prints verbose output to console.

`what` What to look for. Either `peak` to extract spectral information for a certain peak, `rt` to scan by retention time, or `idx` to scan by numeric index. Defaults to "peak" mode.

`...` Additional arguments to plot_spectrum.

Value

If export_spectrum is TRUE, returns the spectra as a data.frame with wavelengths as rows and one column for each sample in the chrom_list encoding the absorbance (or normalized absorbance, if scale_spectrum is TRUE) at each wavelength. Otherwise, there is no return value.

Side effects

If plot_spectrum is TRUE, plots the spectra for the specified chromatogram (chr) of the given peak. The spectrum is a single row from the chromatographic matrix.

Author(s)

Ethan Bass

See Also

plot_spectrum

Examples

```
data(Sa_warp)
pks <- get_peaks(Sa_warp, lambda="220")
pk_tab <- get_peaktable(pks)
plot_all_spectra(peak="V13", peak_table = pk_tab, overlapping=TRUE)
```

---

plot_chroms *Plot traces from list of chromatograms.*

Description

Plots the specified traces from a list of chromatograms.

Usage

```
plot_chroms(
  x,
  lambdas,
  idx,
  xlim,
  ylim,
  xlab = "",
  ylab = "Absorbance",
  engine = c("base", "ggplot", "plotly"),
  linewidth = 1,
  show_legend = TRUE,
  legend_position = "topright",
  ...
)
```

Arguments

`x` A list of chromatograms in matrix format (timepoints x wavelengths).

`lambdas` The wavelength(s) you wish to plot the trace at.

`idx` A vector representing the names or numerical indices of the chromatograms to plot.

`xlim` Range of x axis.

`ylim` Range of y axis.

`xlab` X label.

`ylab` Y label. Defaults to "Absorbance".

`engine` Plotting engine. Either `base` (`matplot`), `plotly`, or `ggplot2-package`.

`linewidth` Line width.

`show_legend` Logical. Whether to display legend or not. Defaults to TRUE.

`legend_position`
Position of legend.

`...` Additional arguments to plotting function specified by `engine`.

Value

No return value, called for side effects.

Side effects

Plots the traces of the specified chromatograms `idx` at the specified wavelengths `lambdas`. Plots can be produced using base graphics, ggplot2, or plotly, according to the value of `engine`.

Author(s)

Ethan Bass

Examples

```
data(Sa_pr)
plot_chroms(Sa_pr, idx=1:2, lambdas=c(210))
```

`plot_spectrum` *Plot spectrum from peak table*

Description

Plots the trace and/or spectrum for a given peak in peak.table object, or plots the spectrum a particular retention time for a given chromatogram.

Usage

```
plot_spectrum(
  loc = NULL,
  peak_table,
  chrom_list,
  idx = "max",
  lambda = "max",
  plot_spectrum = TRUE,
  plot_trace = TRUE,
  spectrum_labels = TRUE,
  scale_spectrum = FALSE,
  export_spectrum = FALSE,
  verbose = TRUE,
  what = c("peak", "rt", "idx", "click"),
  engine = c("base", "plotly", "ggplot2"),
  chr = NULL,
  ...
)
```

Arguments

`loc` The name of the peak or retention time for which you wish to extract spectral data.

`peak_table` The peak table (output from `get_peaktable`).

`chrom_list` A list of chromatograms in matrix format (timepoints x wavelengths). If no argument is provided here, the function will try to find the `chrom_list` object used to create the provided `peak_table`.

`idx` Numerical index of chromatogram you wish to plot, or "max" to automatically plot the chromatogram with the largest signal.

`lambda` The wavelength you wish to plot the trace at if plot_trace == TRUE and/or the wavelength to be used for the determination of signal abundance.

`plot_spectrum` Logical. If TRUE, plots the spectrum of the chosen peak. Defaults to TRUE.

`plot_trace` Logical. If TRUE, plots the trace of the chosen peak at lambda. Defaults to TRUE.

`spectrum_labels`
Logical. If TRUE, plots labels on maxima in spectral plot. Defaults to TRUE.

`scale_spectrum` Logical. If TRUE, scales spectrum to unit height. Defaults to FALSE.

`export_spectrum`
Logical. If TRUE, exports spectrum to console. Defaults to FALSE.

`verbose` Logical. If TRUE, prints verbose output to console. Defaults to TRUE.

`what` What to look for. Either `peak` to extract spectral information for a certain peak, `rt` to scan by retention time, `idx` to scan by numeric index, or `click` to manually select retention time by clicking on the chromatogram. Defaults to "peak" mode.

`engine` Which plotting engine to use: `base`, `ggplot2`, or `plotly`.

`chr` Deprecated. Please use `idx` instead.

`...` Additional arguments.

Details

Can be used to confirm the identity of a peak or check that a particular column in the peak table represents a single compound. Retention times can also be selected by clicking on the plotted trace if what == 'click'.

Value

If `export_spectrum` is TRUE, returns the spectrum as a `data.frame` with wavelengths as rows and a single column encoding the absorbance (or normalized absorbance, if `scale_spectrum` is TRUE) at each wavelength. If `export_spectrum` is FALSE, the output depends on the plotting `engine`. If `engine` == `"plotly"`, returns a `plotly` object containing the specified plots. Otherwise, if `engine` == `"base"`, there is no return value.

Side effects

- If `plot_trace` is TRUE, plots the chromatographic trace of the specified chromatogram (`chr`), at the specified wavelength (`lambda`) with a dotted red line to indicate the retention time given by `loc`. The trace is a single column from the chromatographic matrix.
- If `plot_spectrum` is TRUE, plots the spectrum for the specified chromatogram at the specified retention time. The spectrum is a single row from the chromatographic matrix.

Author(s)

Ethan Bass

Examples

```
data(Sa)
pks <- get_peaks(Sa, lambda="220.00000")
pk_tab <- get_peaktable(pks)
oldpar <- par(no.readonly = TRUE)
par(mfrow=c(2,1))
plot_spectrum(loc = "V10", peak_table = pk_tab, what="peak")
par(oldpar)
```

`preprocess` *Preprocess time/wavelength data*

Description

Standard pre-processing of response matrices, consisting of a time axis and a spectral axis (e.g. HPLC-DAD/UV data). For smooth data, like UV-VIS data, the size of the matrix can be reduced by interpolation. By default, the data are baseline-corrected in the time direction (`baseline.corr`) and smoothed in the spectral dimension using cubic smoothing splines (`smooth.spline`.

Usage

```
preprocess(
  X,
  dim1,
  dim2,
  remove.time.baseline = TRUE,
  spec.smooth = TRUE,
  maxI = NULL,
  parallel = NULL,
  interpolate_rows = TRUE,
  interpolate_cols = TRUE,
  mc.cores,
  cl = 2,
  show_progress = NULL,
  ...
)
```

Arguments

`X` A numerical data matrix, or list of data matrices. Missing values are not allowed. If rownames or colnames attributes are used, they should be numerical and signify time points and wavelengths, respectively.

`dim1` A new, usually shorter, set of time points (numerical). The range of these should not exceed the range of the original time points.

`dim2` A new, usually shorter, set of wavelengths (numerical). The range of these should not exceed the range of the original wavelengths.

`remove.time.baseline` Logical, indicating whether baseline correction should be done in the time direction, according to `baseline.corr`. Default is TRUE.

`spec.smooth` Logical, indicating whether smoothing should be done in the spectral direction, according to `smooth.spline`. Default is TRUE.

`maxI` if given, the maximum intensity in the matrix is set to this value.

`parallel` Logical, indicating whether to use parallel processing. Defaults to TRUE (unless you're on Windows).

`interpolate_rows` Logical. Whether to interpolate along the time axis (`dim1`). Defaults to TRUE.

`interpolate_cols` Logical. Whether to interpolate along the spectral axis (`dim2`). Defaults to TRUE.

`mc.cores` How many cores to use for parallel processing. Defaults to 2. This argument has been deprecated and replaces with `cl`.

`cl` Argument to `pblapply` or `mclapply`. Either an integer specifying the number of clusters to use for parallel processing or a cluster object created by `makeCluster`. Defaults to 2. On Windows integer values will be ignored.

`show_progress` Logical. Whether to show progress bar. Defaults to `TRUE` if `pbapply` is installed.

`...` Further optional arguments to `baseline.corr`.

Value

The function returns the preprocessed data matrix (or list of matrices), with row names and column names indicating the time points and wavelengths, respectively.

Note

Adapted from the preprocess function in the alsace package by Ron Wehrens.

Author(s)

Ethan Bass

References

- Wehrens, R., Bloemberg, T.G., and Eilers P.H.C. 2015. Fast parametric time warping of peak lists. *Bioinformatics* **31**:3063-3065. doi:10.1093/bioinformatics/btv299.
- Wehrens, R., Carvalho, E., Fraser, P.D. 2015. Metabolite profiling in LC–DAD using multivariate curve resolution: the alsace package for R. *Metabolomics* **11:1**:143-154. doi:10.1007/s1130601406835.

Examples

```
data(Sa)
new.ts <- seq(10,18.66,by=.01) # choose time-points
new.lambdas <- seq(200, 318, by = 2) # choose wavelengths
Sa_pr <- preprocess(Sa[[1]], dim1 = new.ts, dim2 = new.lambdas)
```

---

reshape_chroms *Reshape chromatograms Reshapes a list of chromatograms from wide to long format.*

Description

Reshape chromatograms Reshapes a list of chromatograms from wide to long format.

Usage

```
reshape_chroms(x, idx, sample_var = "sample", lambdas = NULL, rts = NULL)
```

Arguments

x	A list of chromatographic matrices in wide format.
idx	Indices of chromatograms to convert
sample_var	String with name of new column containing sample IDs.
lambdas	Vector specifying wavelength(s) to include.
rts	Vector specifying retention times to include.

Value

A list of chromatographic matrices in long format.

Author(s)

Ethan Bass

`reshape_peaktable` *Reshapes peak table from wide to long format*

Description

Reshapes peak table from wide to long format

Usage

```
reshape_peaktable(x, peaks, metadata, fixed_levels = TRUE)
```

Arguments

`x`	A `peak_table` object.
`peaks`	A character vector specifying the peaks to include. If the character vector is named, the names of the vector elements will be used in place of the original peak names.
`metadata`	A character vector specifying the metadata fields to include.
`fixed_levels`	Logical. Whether to fix factor levels of features in the order provided. Defaults to `TRUE`.

Value

A data.frame containing the information for the specified peaks in long format.

Author(s)

Ethan Bass

Sa *Raw goldenrod root chromatograms*

Description

A list of four HPLC-DAD data matrices of *Solidago altissima* roots extracted in 90% methanol. Retention times are stored in rows and wavelengths are stored in columns.

Usage

```
data(Sa)
```

Format

A list of four matrices (1301 times x 60 wavelengths).

Sa_pr *Preprocessed goldenrod root chromatograms*

Description

A list of four pre-processed HPLC-DAD chromatograms derived from the raw data stored in Sa. Retention times are stored in rows and wavelengths are stored in columns. The time axis is compressed to save space and processing time so the data are a little choppy.

Usage

```
data(Sa_pr)
```

Format

A list of four pre-processed matrices (434 retention times x 60 wavelengths).

Sa_warp *Warped goldenrod root chromatograms.*

Description

A list of four pre-processed and warped goldenrod root chromatograms derived from the raw data stored in Sa.

Usage

```
data(Sa_warp)
```

Format

A list of four pre-processed and warped matrices (434 times x 60 wavelengths).

scan_chrom *Plot spectra by clicking on the chromatogram*

Description

Plot spectra by clicking on the chromatogram

Usage

```
scan_chrom(
  chrom_list,
  idx,
  lambda,
  plot_spectrum = TRUE,
  peak_table = NULL,
  scale_spectrum = FALSE,
  spectrum_labels = TRUE,
  export_spectrum = FALSE,
  chr = NULL,
  ...
)
```

Arguments

`chrom_list` A list of chromatograms in matrix format (timepoints x wavelengths). If no argument is provided here, the function will try to find the `chrom_list` object used to create the provided `peak_table`.

`idx` Numerical index of chromatogram you wish to plot.

`lambda` The wavelength to plot the trace at.

`plot_spectrum` Logical. Whether to plot the spectrum or not.

`peak_table` The peak table (output from `get_peaktable` function).

`scale_spectrum` Logical. If TRUE, scales spectrum to unit height. Defaults to FALSE.

`spectrum_labels` Logical. If TRUE, plots labels on maxima in spectral plot. Defaults to TRUE.

`export_spectrum` Logical. If TRUE, exports spectrum to console. Defaults to FALSE.

`chr` Deprecated. Please use `idx` instead.

`...` Additional arguments.

Value

If `export_spectrum` is TRUE, returns the spectrum as a `data.frame` with wavelengths as rows and a single column encoding the absorbance (or normalized absorbance, if `scale_spectrum` is TRUE) at each wavelength. Otherwise, there is no return value.

Side effects

Plots a chromatographic trace from the specified chromatogram (`chr`), at the specified wavelength (`lambda`) with a dotted red line to indicate the user-selected retention time. The trace is a single column from the chromatographic matrix.

If `plot_spectrum` is TRUE, plots the spectrum for the specified chromatogram at the user-specified retention time. The spectrum is a single

Author(s)

Ethan Bass

Examples

```
data(Sa_pr)
scan_chrom(Sa_pr, lambda = "210", idx = 2, export_spectrum = TRUE)
```

---

subset.peak_table *Subset peak table Return subset of* peak_table *object.*

Description

Subset peak table Return subset of `peak_table` object.

Usage

```
## S3 method for class 'peak_table'
subset(x, subset, select, drop = FALSE)
```

Arguments

x A peak_table object

subset Logical expression indicating rows (samples) to keep from peak_table; missing values are taken as false.

select Logical expression indicating columns (peaks) to select from peak_table.

drop Logical. Passed to indexing operator.

Value

A peak_table object with samples specified by subset and peaks specified by select.

Author(s)

Ethan Bass

write_peaktable *Export peak table*

Description

Export peak table

Usage

```
write_peaktable(
  peak_table,
  path,
  filename = "peak_table",
  format = c("csv", "xlsx"),
  what = c("tab", "pk_meta", "sample_meta", "ref_spectra", "args")
)
```

Arguments

peak_table Peak table object from get_peaktable.

path Path to write file.

filename File name. Defaults to "peak_table".

format File format to export. Either csv or xlsx.

what Which elements of the peak_table to export.

Value

No return value. The function is called for its side effects.

Side effects

Exports peak_table object as `.csv` or `.xlsx` file according to the value of `format`.

Examples

```
data(pk_tab)
path_out = tempdir()
write_peaktable(pk_tab, path = path_out, what = c("tab"))
```

Index