

Package: chromatographR (via r-universe)

October 21, 2026

Type Package

Title Chromatographic Data Analysis Toolkit

Version 0.9.0

Maintainer Ethan Bass <ethanbass@gmail.com>

Description Analyzes chromatographic data from high performance liquid chromatography with diode-array or ultraviolet detection (HPLC-DAD/UV), as well as other detectors, in a high-throughput and reproducible workflow. Provides functionality for preprocessing, alignment, peak detection, peak model fitting, peak table construction, and data visualization. Chromatogram alignment is available via parametric time warping (as implemented in 'ptw') or variable penalty dynamic time warping (as implemented in 'VPdtw'). Peak detection is based on a first-derivative zero-crossing algorithm, and peak shapes are fitted using non-linear least squares. Underlying methodologies are described in: Wehrens et al. (2015) <doi:10.1007/s11306-014-0683-5> for preprocessing and peak table construction; Eilers (2004) <doi:10.1021/ac034800e> and Wehrens et al. (2015) <doi:10.1093/bioinformatics/btv299> for parametric time warping; Clifford & Stone (2012) <doi:10.18637/jss.v047.i08> for variable penalty dynamic time warping; Tom O'Haver <<https://terpconnect.umd.edu/~toh/spectrum/PeakFindingandMeasurement.htm>> for peak detection; Lan & Jorgenson (2001) <doi:10.1016/S0021-9673(01)00594-5> for exponential-Gaussian hybrid fitting; and Naish & Hartwell (1988) <doi:10.1007/BF02268168> for bi-exponentially modified Gaussian fitting.

License GPL (>= 2)

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

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

Depends R (>= 4.1.0)

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

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

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website curl, ggdiplots, vegan, quarto

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression xz

Config/roxygen2/version 8.1.1

Roxygen list(markdown = TRUE)

Config/pak/sysreqs cmake make libicu-dev libpng-dev libuv1-dev libxml2-dev python3

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

Date/Publication 2026-10-20 19:36:22 UTC

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

RemoteRef HEAD

RemoteSha 786922fdc7dd2514a151630e466f98a835650f60

Contents

chromatographR-package	3
[.matched_peak_list	6
[.peak_list	7
[.peak_table	8
annotate_peaks	9
attach_metadata	10
attach_ref_spectra	11
baseline_estimators	12
boxplot.peak_table	14
cluster_spectra	15
combine_peaks	17
correct_baseline	19
correct_peaks	21
correct_rt	22
correct_rt_group	26
crop_chroms	30
filter_peaks	31
filter_peaktable	33
get_baseline	34

get_lambdas	35
get_peaks	36
get_peaktable	41
get_spectra	46
get_time_resolution	47
get_times	48
match_peaks	49
merge_peaks	52
mirror_plot	53
normalize_data	55
peak_table-class	56
pk_tab	57
plot.chrom_list	58
plot.matched_peak_list	59
plot.peak_list	60
plot.peak_table	63
plot.ptw_list	65
plot_all_spectra	66
plot_baseline	68
plot_chrom_contour	69
plot_chroms	71
plot_chroms_heatmap	73
plot_spectrum	75
plot_spectrum_inset	77
preprocess	79
print.matched_peak_list	82
print.matched_peak_list_set	83
print.peak_list	83
reshape_chroms	84
reshape_peaktable	85
Sa	86
Sa_pr	87
Sa_warp	87
scan_chrom	88
subtract_blanks	89
targeted_analysis	91
write_peaktable	93
Index	94

Description

Analyzes chromatographic data from high performance liquid chromatography with diode-array or ultraviolet detection (HPLC-DAD/UV), as well as other detectors, in a high-throughput and reproducible workflow. Provides functionality for preprocessing, alignment, peak detection, peak model fitting, peak table construction, and data visualization.

Details

A typical workflow includes signal preprocessing and peak table construction following general approaches described in Wehrens et al. (2015).

Chromatogram alignment is implemented using parametric time warping (PTW) or variable penalty dynamic time warping (VPdtw).

Peak detection is based on zero-crossings in smoothed derivatives. Peaks are then modeled using Gaussian, exponential-Gaussian hybrid, or bidirectional exponentially modified Gaussian peak functions via nonlinear least squares, or integrated directly from the signal.

Further details and example workflows are provided in the package vignettes.

Core analysis functions

- `read_chroms()`: Import chromatograms from a variety of vendor formats.
- `preprocess()`: Preprocess chromatographic matrices.
- `correct_baseline()`: Estimate and subtract a baseline.
- `crop_chroms()`: Restrict chromatograms to a range of times or wavelengths.
- `subtract_blanks()`: Subtract blank chromatograms.
- `correct_rt()`: Align chromatograms.
- `get_peaks()`: Find and integrate peaks.
- `get_peaktable()`: Assemble peak table.
- `attach_metadata()`: Attach metadata to peak table.
- `attach_ref_spectra()`: Attach reference spectra to peak table.
- `normalize_data()`: Normalize `peak_table` or `chrom_list`.
- `get_purity()`: Calculate peak purity.

Peak refinement

- `combine_peaks()`: Combine duplicate peaks based on retention time and spectral similarity.
- `merge_peaks()`: Merge split peaks into a single feature in a peak table.
- `filter_peaktable()`: Filter peak features in a peak table.
- `filter_peaks()`: Filter peaks in peak lists.

Visualization functions**Chromatogram plots:**

- `plot_chroms()`: Plot chromatograms as traces.
- `plot_chroms_heatmap()`: Plot chromatograms as heatmap.
- `annotate_peaks()`: Add peak labels to chromatogram plots.
- `mirror_plot()`: Plot chromatograms as mirror plots.
- `plot_baseline()`: Plot an estimated baseline over its trace.
- `plot.ptw_list()`: Plot parametric time warping (PTW) alignment object.

Spectral plots:

- `plot_spectrum()`: Plot spectrum and/or trace of specified peak.
- `plot_spectrum_inset()`: Plot spectrum over chromatogram.
- `plot_all_spectra()`: Plot all spectra for specified peak.
- `scan_chrom()`: Interactively extract spectra from a chromatogram.

Peak plots:

- `plot.peak_list()`: Plot fitted peaks over chromatographic trace.
- `plot.peak_table()`: Plot chromatograms and/or spectra from a peak table.
- `boxplot.peak_table()`: Create boxplot from a peak table object.

Data utilities and I/O

- `get_times()`: Return retention times from a peak table or a list of chromatograms.
- `get_lambdas()`: Return wavelengths from a peak table or a list of chromatograms.
- `get_time_resolution()`: Return the sampling interval of a chromatogram.
- `get_baseline()`: Retrieve a baseline retained by preprocessing.
- `write_peaktable()`: Export peak table to CSV or XLSX format.
- `reshape_chroms()`: Reshape a list of chromatograms to long format.
- `reshape_peaktable()`: Reshape a peak table to long format.

Example data

- `Sa`: A list of four goldenrod root chromatograms.
- `Sa_pr`: Preprocessed goldenrod root chromatograms.
- `Sa_warp`: Preprocessed and aligned goldenrod root chromatograms.
- `pk_tab`: Peak table from aligned goldenrod root chromatograms.

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. *Analytical Chemistry*, 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

Useful links:

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

[.matched_peak_list] *Subset matched peak list*

Description

Subsets a matched_peak_list while preserving metadata attributes.

Usage

```
## S3 method for class 'matched_peak_list'
x[i, j, ..., drop = FALSE]
```

Arguments

x	A matched_peak_list object.
i	Indices specifying elements to extract.
j	Not used.
...	Additional arguments (ignored).
drop	Not used.

Value

A matched_peak_list object.

Author(s)

Ethan Bass

[.peak_list	<i>Subset peak list</i>
-------------	-------------------------

Description

Subsets a peak_list while preserving metadata attributes.

Usage

```
## S3 method for class 'peak_list'  
x[i, j, ..., drop = FALSE]
```

Arguments

x	A peak_list object.
i	Indices specifying elements to extract.
j	Not used.
...	Additional arguments (ignored).
drop	Not used.

Value

A peak_list object.

Author(s)

Ethan Bass

[.peak_table	<i>Subset peak table</i>
--------------	--------------------------

Description

Subsets a peak_table while preserving and subsetting all associated slots (tab, pk_meta, sample_meta, instrument_meta, ref_spectra).

Usage

```
## S3 method for class 'peak_table'  
x[i, j, ..., drop = FALSE]
```

Arguments

x	A peak_table object.
i	Row indices (samples) to subset.
j	Column indices (peaks) to subset.
...	Additional arguments (ignored).
drop	Passed to the [indexing operator.

Value

A peak_table object.

Note

Subsetting a peak_table does not modify the associated chrom_list. Methods that rely on the chrom_list (e.g., [plot_spectrum](#)) may not work as expected on a subsetting peak_table unless the chrom_list is subsetting separately.

Author(s)

Ethan Bass

annotate_peaks

Annotate peaks on a chromatogram

Description

Adds text labels to peaks on a chromatogram plot produced by [plot_chroms\(\)](#). Peaks named in a peak table are labeled at their retention time, above the highest point of the chromatograms in `idx` between the peak's start and end. Peaks given as retention times are labeled at that time, above the highest plotted point within `tol` of it, and need no peak table; the label then marks the tallest trace in that window, whichever peak it belongs to.

Usage

```
annotate_peaks(
  p,
  loc,
  peak_table = NULL,
  chrom_list = NULL,
  label = NULL,
  lambda = NULL,
  idx = NULL,
  tol = 0.05,
  vjust = -0.5,
  ...
)
```

Arguments

<code>p</code>	A ggplot object produced by plot_chroms() .
<code>loc</code>	Either a character vector of peak names from <code>peak_table</code> (e.g. <code>c("V8", "V15")</code>) or a numeric vector of retention times, in the units of the plot's time axis. If named, the names are used as labels.
<code>peak_table</code>	A <code>peak_table</code> object created by get_peaktable . Required when <code>loc</code> gives peak names.
<code>chrom_list</code>	A list of chromatograms in matrix format (timepoints × wavelengths). If no argument is provided here, the function will try to find the <code>chrom_list</code> object using the pointer in the <code>peak_table</code> .
<code>label</code>	Labels to display at each peak. Can be a character vector of labels (one per peak), or <code>"rt"</code> to use the retention times. If labels are not specified here, the names of <code>loc</code> will be applied, or if the <code>loc</code> vector is not named, the peak names will be used directly.
<code>lambda</code>	Wavelength(s) to use for locating the peak apex. Inherited from <code>p</code> if <code>NULL</code> (default). If multiple wavelengths are provided, the one with the highest absorbance at the peak's retention time is used.

idx	Index of chromatogram(s) to use for locating the peak apex. Inherited from p if NULL (default). lambda and idx apply only to peak names; retention times use the plotted traces.
tol	Half-width of the window around each retention time in which the highest plotted point is found, in minutes. Defaults to 0.05. Ignored for peak names.
vjust	Vertical justification of the label relative to the peak apex. Defaults to -0.5.
...	Additional arguments passed to <code>ggplot2::annotate()</code> .

Value

A ggplot object.

See Also

`plot_chroms()`, `plot_spectrum_inset()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  data(Sa_warp)
  data(pk_tab)
  plot_chroms(Sa_warp, lambdas = 210, engine = "ggplot") |>
    annotate_peaks(c(C1="V7", C2="V9", C3="V15"), peak_table = pk_tab)
  plot_chroms(Sa_warp, lambdas = 210, engine = "ggplot") |>
    annotate_peaks(c(C1 = 12.18, C2 = 12.70))
}
```

attach_metadata	<i>Attach sample metadata to a peak_table</i>
-----------------	---

Description

Matches rows of metadata to samples by name and stores them in the `sample_meta` slot, in the same order as the rows of `peak_table$tab`. Samples missing from metadata get NA values, with a warning. Columns already in `sample_meta`, such as fields added to the chromatograms with `chromConverter::add_metadata()` (`chromConverter` 0.11.0 or later), are kept unless metadata replaces them. To replace `sample_meta` entirely, set `peak_table$sample_meta <- NA` first.

Usage

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

Arguments

peak_table	A peak_table object created by <code>get_peaktable</code> .
metadata	A data.frame (or tibble) of sample metadata.
column	Name of the column in metadata containing sample identifiers, which must be unique and match the row names of <code>peak_table\$tab</code> .

Value

A peak_table object with metadata stored 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	<i>Attach reference spectra to a peak_table object</i>
--------------------	--

Description

For each peak, selects one sample's spectrum as the reference: either the spectrum with the highest mean correlation to that peak's spectra in the other samples ("max.cor"), or the spectrum from the sample where the peak is most intense ("max.int"). Spectra are scaled before they are stored.

Usage

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

Arguments

peak_table	A peak_table object created by get_peaktable .
chrom_list	Optional list of chromatograms (timepoints × wavelengths). If omitted, the function attempts to retrieve the original chrom_list used to generate peak_table.
ref	Method for selecting reference spectra: "max.cor" (maximum mean correlation, the default) or "max.int" (maximum intensity).

Value

A peak_table object with reference spectra stored in the \$ref_spectra slot.

Author(s)

Ethan Bass

See Also

[get_peaks](#), [get_peaktable](#)

Examples

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

baseline_estimators *Baseline estimators*

Description

Reference for the baseline estimators available through the `baseline` argument of [preprocess](#) and [correct_baseline](#).

Details

A baseline is estimated separately for each wavelength, since the drift differs across the spectrum. Which estimator performs best depends on the noise, the drift, and how crowded the trace is, and a smoothing parameter that suits one chromatogram need not suit another. Rather than choosing on general grounds, fit a baseline and look at it: set `attach_baseline = TRUE` in [preprocess](#) or [correct_baseline](#) and plot the estimate with [plot_baseline](#), as in the examples below.

Estimator	Parameters (defaults)	Asymmetry	Overshoots	Scope
"als" (default)	stiffness (1e7), p (0.001)	fixed	yes	global
"arpls"	stiffness (1e7)	adaptive	yes	global
"aspls"	stiffness (1e9), k (0.5)	adaptive	yes	global
"snip"	m (required)	fixed	no	local

- "als" is asymmetric least squares, as implemented by [asysm](#). A Whittaker smoother is fit to the signal with residuals above the fit weighted by p and residuals below it by $1 - p$. p is fixed for the whole trace, so the asymmetry is set in advance rather than taken from the data.
- "arpls" is asymmetrically reweighted penalized least squares (Baek et al. 2015). It uses the same Whittaker solve, but replaces the fixed p with weights computed at each iteration from the mean and standard deviation of the negative residuals, so the asymmetry follows the trace. On very clean traces it can fail to follow steep drift, and the baseline falls below the trace where the drift is steepest.
- "aspls" is adaptive smoothness parameter penalized least squares (Zhang et al. 2020). It keeps the re-weighting and additionally scales the penalty at each timepoint by the normalized absolute residual, so the fit is stiffest where residuals are largest and most flexible where they are smallest. Since that scaling is at most one, it lowers the effective penalty. Accordingly, the default stiffness is set 100 times higher than the other two estimators and a value tuned for "als" or "arpls" will under-smooth. The best conversion varies by trace, so retune stiffness rather than rescaling it.

- "snip" is Statistics-sensitive Non-linear Iterative Peak-clipping (Ryan et al. 1988). Its baseline follows variation broader than its $2m + 1$ timepoint window and passes beneath narrower features, such as peaks. Two of its properties follow from its construction: the estimate can never exceed the signal, so the corrected trace is non-negative; and each point depends only on the $2m + 1$ window around it, so a distortion in one part of the trace cannot propagate along it the way it can through a global solve.

"als" and "snip" are both one-sided: the first weights positive residuals down by a fixed factor, the second cannot exceed the signal at all. Both therefore fit below the center of the noise, which leaves the corrected trace biased high. The bias is a constant offset in signal units, so it inflates small peaks proportionally more than large ones. "arpls" and "aspls" take their asymmetry from the residuals instead and fit closer to the center of the noise.

Parameters

Parameters are passed to the estimator through the ... of [preprocess](#) or [correct_baseline](#).

- stiffness ("als", "arpls", "aspls") is the smoothing penalty. Larger values give a stiffer, smoother baseline. It acts between adjacent timepoints, so the same value spans a different stretch of time on a differently sampled trace; retune it if you change the sampling interval. The defaults are starting points, not calibrated optima.
- p ("als") is the weight given to points above the fit, and $1 - p$ the weight given to points below it. At the default of 0.001, points above the fit, which are mostly peaks, barely pull on it.
- k ("aspls") sets the steepness of the weighting curve. Larger values switch more abruptly between points treated as baseline and points treated as peak.
- m ("snip") is the half-width of the clipping window, in timepoints. It has no default: the window has to be wider than the peaks it clips, which depends on your chromatography.

Acknowledgment

The "aspls" implementation follows pybaselines (Erb 2025) at two points that Zhang et al. leave open: the default asymmetric coefficient k is 0.5 rather than the 2 printed in the paper, and the second-difference matrix is left-multiplied by the diagonal matrix of per-timepoint penalty scalings, an ordering the paper's notation does not settle.

Author(s)

Ethan Bass

References

- Baek, S.-J., Park, A., Ahn, Y.-J., and Choo, J. 2015. Baseline correction using asymmetrically reweighted penalized least squares smoothing. *Analyst* 140:250-257. doi:10.1039/C4AN01061B.
- Zhang, F., Tang, X., Tong, A., Wang, B., Wang, J., Lv, Y., Tang, C., and Wang, J. 2020. Baseline correction for infrared spectra using adaptive smoothness parameter penalized least squares method. *Spectroscopy Letters* 53:222-233. doi:10.1080/00387010.2020.1730908.

- Ryan, C.G., Clayton, E., Griffin, W.L., Sie, S.H., and Cousens, D.R. 1988. SNIP, a statistics-sensitive background treatment for the quantitative analysis of PIXE spectra in geoscience applications. *Nuclear Instruments and Methods in Physics Research B* 34:396-402. doi:10.1016/0168583X(88)900638.
- Erb, D. 2025. *pybaselines: A Python library of algorithms for the baseline correction of experimental data*. doi:10.5281/zenodo.5608581.

See Also

[preprocess](#), [correct_baseline](#), [plot_baseline](#), [get_baseline](#)

Examples

```
data(Sa)
als <- preprocess(Sa[1], dim1 = seq(10, 18, by = .01),
                  dim2 = seq(200, 318, by = 2), attach_baseline = TRUE,
                  show_progress = FALSE, cl = 1)
snip <- preprocess(Sa[1], dim1 = seq(10, 18, by = .01),
                  dim2 = seq(200, 318, by = 2), baseline = "snip", m = 50,
                  attach_baseline = TRUE, show_progress = FALSE, cl = 1)
plot_baseline(als, lambda = 210, annotate = FALSE)
plot_baseline(snip, lambda = 210, col = "blue", add = TRUE)
legend("topright", legend = c("als", "snip"), col = c("red", "blue"),
      lwd = 2)
```

boxplot.peak_table	<i>Make boxplot from peak table</i>
--------------------	-------------------------------------

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 .

Value

No return value, called for side effects.

Side effects

Creates a boxplot according to the provided formula, using data from the supplied `peak_table` object.

Author(s)

Ethan Bass

See Also

Other visualization functions: `mirror_plot()`, `plot.chrom_list()`, `plot.matched_peak_list()`, `plot.peak_list()`, `plot.peak_table()`, `plot_all_spectra()`, `plot_chrom_contour()`, `plot_chroms()`, `plot_chroms_heatmap()`, `plot_spectrum()`, `scan_chrom()`

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=V9 ~ trt)
```

cluster_spectra	<i>Cluster spectra</i>
-----------------	------------------------

Description

Cluster peaks by spectral similarity.

Usage

```
cluster_spectra(
  peak_table,
  alpha = 0.05,
  min_size = 5,
  max_size = NULL,
  nboot = 1000,
  plot_dend = TRUE,
  plot_spectra = TRUE,
  verbose = getOption("verbose"),
  outfile = NULL,
  save = NULL,
  parallel = TRUE,
  max.only = FALSE,
  output = c("pvclust", "clusters"),
  col = palette.colors(palette = "Okabe-Ito")[-c(1, 5, 9)],
  ...
)
```

Arguments

peak_table	A peak_table object from get_peaktable .
alpha	Significance threshold used to select clusters based on approximately unbiased (AU) bootstrap p-values from pvclust .
min_size	Clusters must contain more than min_size peaks. Defaults to 5.
max_size	Clusters must contain fewer than max_size peaks. Defaults to the number of peaks in peak_table.
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.
outfile	Optional path to an .RDS file. If not NULL, the pvclust object is saved to file using <code>saveRDS()</code> . Existing files are overwritten.
save	The save argument is deprecated. Please use outfile instead.
parallel	Logical. If TRUE, use parallel processing for pvclust .
max.only	Logical. If TRUE, returns only highest level cluster from each nested set of clusters. Named to match the argument of the same name in pvrect and pvpick ; passed to <code>pvpick</code> .
output	What to return: "pvclust" for the pvclust object, "clusters" for the list of clusters, or both (the default).
col	Colors for each cluster's box and number on the dendrogram and the title of its spectra panel, recycled if there are more clusters than colors. Defaults to the Okabe-Ito palette, without black, yellow and grey.
...	Additional arguments to pvclust .

Details

Before using this function, reference spectra must be attached to the peak_table using the [attach_ref_spectra](#) function. These spectra are then used to construct a distance matrix from pairwise Pearson correlations among peaks, and [pvclust](#) clusters the peaks hierarchically, with bootstrap resampling. Clusters whose bootstrap support exceeds the confidence set by alpha are returned. Peaks whose reference spectrum is constant are dropped before clustering.

Clusters can be filtered by size using the min_size and max_size arguments. The clustering algorithm often returns nested clusters, so an individual peak can appear in more than one. If max.only is TRUE, only the largest cluster in a nested tree of clusters meeting the confidence threshold is returned.

The example uses nboot = 100 to reduce run time. For real data, the pvclust authors recommend the default of 1000 replicates for a first look, then 10,000 for smaller errors.

Value

Returns a named list with an element for each value of output: pvclust, the pvclust object, and clusters, a 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 scalar containing one minus the approximately unbiased (AU) value that **pvclust** reports for the given cluster, so that small values indicate strong support.

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](https://doi.org/10.1093/bioinformatics/btl117).

Examples

```
if (requireNamespace("pvclust", quietly = TRUE)) {  
  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, alpha = 0.03)  
}
```

combine_peaks

Combine duplicate peaks

Description

Identifies groups of duplicate peaks in a peak table and retains a single representative peak from each group. Peaks are considered duplicates when their retention times differ by less than `tol` and their spectral correlation exceeds `min_cor`. Use it to collapse a compound that was integrated at more than one wavelength into a single feature.

Usage

```
combine_peaks(  
  peak_table,  
  tol = 0.01,  
  min_cor = 0.9,  
  choose = c("max", "least_sparse", "lambda"),  
  lambda = NULL,  
  verbose = getOption("verbose"),  
  min.cor = NULL  
)
```

Arguments

peak_table	Peak table from get_peaktable .
tol	Tolerance for matching retention times (maximum retention time difference), in minutes. Defaults to 0.01.
min_cor	Minimum spectral correlation to confirm a match. Defaults to 0.9.
choose	Method used to select a representative peak from each group of duplicate peaks. "max" retains the peak with the greatest total intensity, "least_sparse" retains the peak detected in the greatest number of samples, and "lambda" retains peaks matching the preferred wavelength(s).
lambda	Character vector of preferred wavelength(s) to retain when choose = "lambda". The function keeps the duplicate peak whose integration wavelength matches one of the supplied values; if several match, it keeps the first in the peak table, whatever the order of lambda. A group with no match falls back to choose = "max". An error is thrown if none of the supplied values match any peak in the peak table.
verbose	Logical. Whether to print status to the console.
min.cor	The min.cor argument is deprecated. Please use min_cor instead.

Value

A peak table derived from the original, but with columns corresponding to duplicate peaks combined according to the specified criteria.

Author(s)

Ethan Bass

See Also

[get_peaks](#)

Other utility functions: [merge_peaks\(\)](#), [reshape_chroms\(\)](#), [reshape_peaktable\(\)](#)

Examples

```
data(pk_tab)
data(Sa_warp)
pk_tab <- attach_ref_spectra(pk_tab)
combine_peaks(pk_tab, tol = 0.02, min_cor = 0.9)
```

correct_baseline	<i>Correct baseline along the time axis</i>
------------------	---

Description

Estimates a baseline along the time axis of each chromatogram and subtracts it, using any of the estimators described in [baseline_estimators](#).

Usage

```
correct_baseline(
  x,
  baseline = c("als", "arpls", "aspls", "snip"),
  attach_baseline = FALSE,
  zero_floor = FALSE,
  cl = 2,
  show_progress = NULL,
  ...
)
```

Arguments

<code>x</code>	A numeric matrix (time × wavelength), a list of such matrices, or a plain numeric vector for a single trace. Row names (or, for a vector, names) must correspond to time points and column names to wavelengths.
<code>baseline</code>	Baseline estimator to use along the time axis. Either asymmetric least squares ("als", the default, via asysm), asymmetrically reweighted penalized least squares ("arpls"), adaptive smoothness parameter penalized least squares ("aspls"), or statistics-sensitive non-linear iterative peak-clipping ("snip"). See baseline_estimators for a description of each.
<code>attach_baseline</code>	Logical. Whether to retain the estimated baseline alongside the corrected data, as an "estimated_baseline" attribute on each returned matrix, retrievable with get_baseline . The baseline is subtracted either way; this argument only controls whether the estimate is kept for inspection. Defaults to FALSE: the baseline is estimated separately for each wavelength, so it is the same size as the data and retaining it doubles memory use.
<code>zero_floor</code>	Logical. Whether to clip negative values in the corrected data to zero. Defaults to FALSE, because clipping also removes the negative dips that show where a baseline was over-subtracted. It can be useful with <code>baseline = "als"</code> , which can overshoot the signal; "snip" cannot overshoot by construction, so clipping is never needed there. The <code>zero_floor</code> argument of subtract_blanks , by contrast, defaults to TRUE, since a negative value there means the blank exceeded the sample rather than that an estimator overshoot.
<code>cl</code>	Either an integer specifying the number of cores to use for parallel processing or a cluster object created by makeCluster . Defaults to 2. On Windows, integer values are ignored.

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

... Additional arguments to the selected baseline estimator: e.g. `stiffness` and `p` for "als" (`asysm`), `stiffness` for "arpls", `stiffness` and `k` for "aspls", or `m` for "snip".

Details

Unlike `preprocess`, this function only subtracts the baseline; it does not interpolate, smooth, or rescale.

Value

The baseline-corrected data, in the same shape as `x`, with row and column names (or, for a vector, names) unchanged.

Author(s)

Ethan Bass

References

- Baek, S.-J., Park, A., Ahn, Y.-J., and Choo, J. 2015. Baseline correction using asymmetrically reweighted penalized least squares smoothing. *Analyst* 140:250-257. doi:10.1039/C4AN01061B.
- Zhang, F., Tang, X., Tong, A., Wang, B., Wang, J., Lv, Y., Tang, C., and Wang, J. 2020. Baseline correction for infrared spectra using adaptive smoothness parameter penalized least squares method. *Spectroscopy Letters* 53:222-233. doi:10.1080/00387010.2020.1730908.
- Ryan, C.G., Clayton, E., Griffin, W.L., Sie, S.H., and Cousens, D.R. 1988. SNIP, a statistics-sensitive background treatment for the quantitative analysis of PIXE spectra in geoscience applications. *Nuclear Instruments and Methods in Physics Research B* 34:396-402. doi:10.1016/0168583X(88)900638.
- Erb, D. 2025. *pybaselines: A Python library of algorithms for the baseline correction of experimental data*. doi:10.5281/zenodo.5608581.

See Also

`baseline_estimators`, `preprocess`, `get_baseline`, `subtract_blanks`

Other preprocessing functions: `crop_chroms()`, `preprocess()`, `subtract_blanks()`

Examples

```
data(Sa_pr)
bc <- correct_baseline(Sa_pr[1], cl = 1, show_progress = FALSE)
```

correct_peaks	<i>Correct peak positions according to a warping model</i>
---------------	--

Description

Corrects retention time differences in `peak_list` using warping models fitted by `correct_rt` with any of its algorithms.

Usage

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

Arguments

<code>peak_list</code>	A <code>peak_list</code> object created by <code>get_peaks</code> , containing a nested list of peak tables where the first level is the sample, and the second level is the spectral wavelength. Every wavelength is described by a matrix where each row corresponds to a feature, and the columns contain information on that feature (e.g., retention time, peak width (FWHM), height, area, etc.)
<code>mod_list</code>	Warping models, as returned by <code>correct_rt</code> with <code>what = "models"</code> : a <code>ptw_list</code> , an <code>lptw_list</code> , or a <code>VPdtw</code> object. Corrected times are on the time axis of the chromatograms <code>correct_rt</code> returns, which for "vpdtw" extends beyond the reference at either end.
<code>chrom_list</code>	List of chromatograms from which the models are derived. If omitted, it is retrieved using the name stored in <code>mod_list</code> .
<code>match_names</code>	Logical. If TRUE (default), models are matched to the samples of <code>peak_list</code> by name when both are named; otherwise by position.
<code>chrom_list_aligned</code>	Optional aligned chromatograms, as returned by <code>correct_rt</code> with <code>what = "corrected.values"</code> . Only its name is recorded, so the object must still exist under that name when it is used.

Details

Once an appropriate warping model has been established, corrected retention times can be predicted for each peak. These are stored in an `rt.cor` column, alongside `start.cor` and `end.cor` when the peak tables have `start` and `end` columns.

The corrected times refer to the aligned time axis, so functions that later read spectra or traces at those times need the aligned chromatograms. The returned peak list stores the warping models for this purpose, and `get_peaktable` carries them into the peak table, where they are re-applied to

the unaligned chromatograms when needed. To use a particular aligned object instead, pass it as `chrom_list_aligned`.

Value

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

Note

This function is adapted from the `correctPeaks` function in the `alsace` package by Ron Wehrens: <https://github.com/rwehrens/alsace/blob/master/R/correctPeaks.R>.

Author(s)

Ron Wehrens, Ethan Bass

See Also

[correct_rt](#)

Examples

```
data(Sa_pr)
warp <- correct_rt(chrom_list = Sa_pr, lambdas = 210, what = "models")
pks <- get_peaks(Sa_pr, lambda = "210")
correct_peaks(pks, mod_list = warp, chrom_list = Sa_pr)
```

correct_rt

Retention time alignment via time warping

Description

Chromatograms are aligned using one of three algorithms, according to the value of `alg`: parametric time warping ("ptw"), as implemented in [ptw](#); variable penalty dynamic time warping ("vpdtw"), as implemented in [VPdtw](#); or landmark parametric time warping ("lptw"), which fits the warping polynomial directly to matched peak positions rather than full chromatographic signals.

Usage

```
correct_rt(
  chrom_list,
  lambdas,
  models = NULL,
  reference = "best",
  alg = c("ptw", "lptw", "vpdtw"),
  peaks = NULL,
  what = c("corrected.values", "models"),
  init.coef = c(0, 1, 0),
```

```

    n_traces = NULL,
    fill_zeros = FALSE,
    n_zeros = 0,
    scale = FALSE,
    trwdth = 0.2,
    plot_it = FALSE,
    penalty = 0.2,
    max_shift = 1,
    dilation_span = 0.3,
    rt_window = 0.3,
    spectral_cutoff = 0.95,
    regularization = 0,
    verbose = getOption("verbose"),
    show_progress = NULL,
    cl = 2,
    ...,
    n.traces = NULL,
    n.zeros = NULL,
    maxshift = NULL
)

```

Arguments

chrom_list	List of chromatograms in matrix format, sharing a common time axis (see preprocess).
lambdas	A character or numeric vector specifying the wavelengths to use for alignment. "vpdtw" and "lptw" accept only a single wavelength and raise an error otherwise. For one-dimensional chromatograms, this argument can be omitted.
models	List of models to warp by. The models provided here (if any) must match the algorithm selected in alg, and are matched to the samples of chrom_list by name.
reference	Index or name of the sample to be used as the reference. Fitted models record it by name. If not specified, chosen by bestref from a similarity matrix of the chromatograms.
alg	Alignment algorithm to use: parametric time warping ("ptw"), landmark parametric time warping ("lptw"), or variable penalty dynamic time warping ("vpdtw").
peaks	A peak list object from get_peaks or a matched_peak_list object from match_peaks . If a matched_peak_list is provided, peak matching will be skipped. Only used when alg = "lptw". If NULL, peaks are found with get_peaks and matched with match_peaks , both at their default settings. To change either, pass their output instead.
what	Output type: either the "corrected.values" (useful for visual inspection and downstream analysis) or the warping "models" (for further programmatic use).
init.coef	Starting values for the optimization. Named to match the argument of the same name in ptw , which receives it. For alg = "lptw", only its length is used, setting the degree of the warping polynomial.

n_traces	Number of wavelengths to warp by, chosen from <code>lambdas</code> (or from all wavelengths if <code>lambdas</code> is missing) by the CODA criterion in <code>select.traces</code> . Defaults to <code>NULL</code> , which uses every wavelength in <code>lambdas</code> .
fill_zeros	Logical. If <code>TRUE</code> , out-of-bounds regions produced by warping are filled with zeros; otherwise they are returned as <code>NA</code> (default).
n_zeros	Number of zeros to pad onto each end of the chromatograms before fitting. Only used when <code>alg = "ptw"</code> .
scale	Logical. If <code>TRUE</code> , divide each chromatogram by its largest absolute value before fitting the warp, so that samples of very different concentration are matched on the shape of their signal. The corrected values are returned in their original units; to rescale the data themselves, use the <code>maxI</code> argument of <code>preprocess</code> . <code>"ptw"</code> is unaffected under its default criterion, weighted cross-correlation, which does not change when a sample is multiplied by a constant.
trwdth	Width of the triangle in the weighted cross-correlation (WCC) criterion, passed to <code>ptw</code> . Defaults to <code>0.2</code> , which matches the default of <code>20</code> timepoints in <code>ptw</code> itself on a chromatogram sampled every <code>0.01</code> minutes. Roughly a peak width is a reasonable choice.
plot_it	Logical. Whether to plot alignment. Defaults to <code>FALSE</code> .
penalty	Numeric ≥ 0 . Strength of the penalty against warping for <code>alg = "vpdtw"</code> : the constant of proportionality in $\lambda \propto \text{dilation}(\text{reference})$ over a <code>dilation_span</code> -wide window, as recommended by Clifford and Stone (2012). A higher number allows less warping; <code>0</code> applies no penalty at all. Defaults to <code>0.2</code> . In <code>chromatographR <= 0.8.0</code> <code>penalty</code> was a divisor, so use the reciprocal of an old value. Values above <code>1</code> raise a warning, since they are more likely to be old divisors than deliberate choices.
max_shift	Maximum allowable shift for <code>VPdtw</code> warping, in minutes. Set it from the retention time drift you actually expect; the default of <code>1</code> allows a minute of drift. Although <code>max_shift</code> is a time, <code>VPdtw</code> itself works on the index axis, so the shift element of the returned models is reported in timepoints.
dilation_span	Width of the moving-maximum window used to build the <code>"vpdtw"</code> penalty. The penalty at each point is the largest reference intensity within <code>dilation_span</code> either side of it, computed by <code>dilation</code> after conversion to timepoints. Smaller values track individual peaks more closely, allowing more warping in the baseline between them. Larger values approach a constant penalty, which becomes exactly constant once <code>dilation_span</code> exceeds the distance between the tallest peak of the reference and the furthest end of the chromatogram. Defaults to <code>0.3</code> , around a couple of peak widths on the example data.
rt_window	Numeric. Argument to <code>match_peaks</code> . Maximum allowed difference in retention time, in minutes, between a reference and sample peak to be considered a candidate match. Defaults to <code>0.3</code> .
spectral_cutoff	Numeric. Argument to <code>match_peaks</code> . Minimum spectral correlation for a candidate peak pair to be retained. Defaults to <code>0.95</code> .
regularization	Numeric ≥ 0 . Strength of the penalty against warping for <code>alg = "lptw"</code> : a dimensionless weight pulling the warp toward the identity warp, passed as <code>penalty</code>

	to <code>fit_penalized_warp</code> . A higher number allows less warping; 0 (the default) applies no penalty at all. 1 weights a unit of squared warp displacement equally against a unit of squared peak-position mismatch, and around 2.4 halves the applied warp, so useful values run roughly 0-10. Unbounded above, approaching the identity warp asymptotically.
<code>verbose</code>	Logical. Whether to print a progress message as the warping models are fit and applied. Defaults to <code>getOption("verbose")</code> .
<code>show_progress</code>	Logical. Whether to show progress bar. Defaults to TRUE if <code>pbapply</code> is installed. Has no effect when <code>alg = "vpdtw"</code> .
<code>cl</code>	Either an integer specifying the number of cores to use for parallel processing or a cluster object created by <code>makeCluster</code> . Defaults to 2. On Windows systems, integers are ignored. Has no effect when <code>alg = "vpdtw"</code> .
<code>...</code>	For <code>alg = "ptw"</code> , optional additional arguments passed to <code>ptw</code> . The only argument that cannot be changed is <code>warp.type</code> , which is hard-coded to "global" to permit warping on multiple wavelengths. For <code>alg = "lptw"</code> , optional arguments passed to <code>fit_penalized_warp</code> : <code>penalty_scale</code> (per-coefficient penalty multipliers) and <code>min_peaks</code> (minimum matched peaks required to fit the polynomial). Its <code>penalty</code> argument is set by regularization rather than through <code>...</code>
<code>n.traces</code>	The <code>n.traces</code> argument is deprecated. Please use <code>n_traces</code> instead.
<code>n.zeros</code>	The <code>n.zeros</code> argument is deprecated. Please use <code>n_zeros</code> instead.
<code>maxshift</code>	The <code>maxshift</code> argument is deprecated. Please use <code>max_shift</code> , which is in minutes. <code>maxshift</code> is read as a count of timepoints, as in <code>chromatographR</code> 0.8.0, and converted.

Details

Some arguments are specific to particular warping functions. For example `n_traces` applies only to "ptw" warping; `penalty`, `max_shift`, and `dilation_span` apply only to "vpdtw" warping; and `peaks`, `rt_window`, `spectral_cutoff`, and `regularization` apply only to "lptw" warping.

`trwdth`, `max_shift` and `dilation_span` are widths along the time axis, given in minutes, regardless of the time unit of `chrom_list`, and rounded internally to a whole number of timepoints. `trwdth` was a count of timepoints up to `chromatographR` 0.8.0; multiply an old value by the sampling interval in minutes to convert it. A width wider than the chromatogram itself is rejected with an error, since it is almost certainly a stale timepoint count. `max_shift` replaces `maxshift`, which is still accepted as a count of timepoints.

`regularization` ("lptw") and `penalty` ("vpdtw") are not comparable: both are dimensionless and higher means less warping, but they are on unrelated scales.

Value

A list of warping models or a list of warped absorbance profiles, according to the value of the `what` argument. With `alg = "vpdtw"`, the warped chromatograms share a new time axis, which can be longer than the input.

Note

The PTW branch of `correct_rt` is adapted from the `correctRT` function in Ron Wehrens' `alsace` package.

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. *Analytical Chemistry*, 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 multi-variate curve resolution: the alsace package for R. *Metabolomics*, 11:143-154. doi:10.1007/s1130601406835.

See Also

[correct_rt_group](#), [correct_peaks](#), [ptw](#), [VPdtw](#)

Examples

```
data(Sa_pr)
warp <- correct_rt(chrom_list = Sa_pr, lambdas=210)
```

`correct_rt_group`

Correct retention time with group-based warping

Description

Aligns chromatograms using parametric time warping ("ptw"), as implemented in [ptw](#), or landmark parametric time warping ("lptw"), with warping functions estimated from within-group averages. This is useful when samples fall into batches with shared retention time shifts, or when individual samples contain peaks absent in other groups that would otherwise confound alignment. One warp is fit per group rather than one per sample, so the cost of warping falls roughly in proportion to the number of groups.

Usage

```

correct_rt_group(
  chrom_list,
  lambdas,
  groups,
  reference = "best",
  reference_group = NULL,
  alg = c("ptw", "lptw"),
  init.coef = c(0, 1, 0),
  n_traces = NULL,
  fill_zeros = FALSE,
  n_zeros = 0,
  scale = FALSE,
  trwidth = 0.2,
  plot_it = FALSE,
  penalty = NULL,
  max_shift = 1,
  rt_window = 0.3,
  spectral_cutoff = 0.95,
  regularization = 0,
  verbose = getOption("verbose"),
  show_progress = NULL,
  cl = 2,
  what = c("corrected.values", "models"),
  ...,
  n.traces = NULL,
  n.zeros = NULL,
  maxshift = NULL
)

```

Arguments

chrom_list	A list of chromatograms in matrix format, sharing a common time axis (see preprocess).
lambdas	A character or numeric vector specifying the wavelengths to use for alignment.
groups	A vector of group assignments for each chromatogram, or a single string naming a metadata attribute from which to extract the group assignments (e.g. "batch"). If a vector is provided, it must be the same length as chrom_list and may be named (matching names(chrom_list)) or positional. All samples must have a group assignment; NA values will trigger an error.
reference	Index or name of a sample in chrom_list whose group will be used as the alignment reference. Defaults to "best", which selects the reference group automatically using bestref applied to the group averages.
reference_group	Name of the group to use as the reference group. If supplied, overrides reference. Defaults to NULL.

alg	Alignment algorithm to use: parametric time warping ("ptw", the default) or landmark parametric time warping ("lptw").
init.coef	Starting values for the optimization. Named to match the argument of the same name in ptw , which receives it. For alg = "lptw", only its length is used, setting the degree of the warping polynomial.
n_traces	Number of wavelengths to warp by, chosen from lambdas (or from all wavelengths if lambdas is missing) by the CODA criterion in select.traces . Defaults to NULL, which uses every wavelength in lambdas .
fill_zeros	Logical. If TRUE, out-of-bounds regions produced by warping are filled with zeros; otherwise they are returned as NA (default).
n_zeros	Number of zeros to pad onto each end of the chromatograms before fitting. Only used when alg = "ptw".
scale	Logical. If TRUE, divide each chromatogram by its largest absolute value before fitting the warp, so that samples of very different concentration are matched on the shape of their signal. The corrected values are returned in their original units; to rescale the data themselves, use the maxI argument of preprocess . "ptw" is unaffected under its default criterion, weighted cross-correlation, which does not change when a sample is multiplied by a constant.
trwdth	Width of the triangle in the weighted cross-correlation (WCC) criterion, passed to ptw . Defaults to 0.2, which matches the default of 20 timepoints in ptw itself on a chromatogram sampled every 0.01 minutes. Roughly a peak width is a reasonable choice.
plot_it	Logical. Whether to plot alignment. Defaults to FALSE.
penalty	Unused. correct_rt_group does not support "vpdtw", so the "vpdtw" penalty has no effect here.
max_shift	Unused. correct_rt_group does not support "vpdtw", so the "vpdtw" shift limit has no effect here.
rt_window	Numeric. Argument to match_peaks . Maximum allowed difference in retention time, in minutes, between a reference and sample peak to be considered a candidate match. Defaults to 0.3.
spectral_cutoff	Numeric. Argument to match_peaks . Minimum spectral correlation for a candidate peak pair to be retained. Defaults to 0.95.
regularization	Numeric >= 0. Strength of the penalty against warping for alg = "lptw": a dimensionless weight pulling the warp toward the identity warp, passed as penalty to fit_penalized_warp . A higher number allows less warping; 0 (the default) applies no penalty at all. 1 weights a unit of squared warp displacement equally against a unit of squared peak-position mismatch, and around 2.4 halves the applied warp, so useful values run roughly 0-10. Unbounded above, approaching the identity warp asymptotically.
verbose	Logical. Whether to print a progress message as the warping models are fit and applied. Defaults to getOption("verbose") .
show_progress	Logical. Whether to show progress bar. Defaults to TRUE if pbapply is installed. Has no effect when alg = "vpdtw".

cl	Either an integer specifying the number of cores to use for parallel processing or a cluster object created by makeCluster . Defaults to 2. On Windows systems, integers are ignored. Has no effect when alg = "vpdtw".
what	What to return: "corrected.values" (the default) for the warped chromatograms, or "models" for a ptw_list or lptw_list holding each sample's group warp, which can be passed to correct_peaks or to correct_rt via models.
...	For alg = "ptw", optional additional arguments passed to ptw . The only argument that cannot be changed is warp.type, which is hard-coded to "global" to permit warping on multiple wavelengths. For alg = "lptw", optional arguments passed to fit_penalized_warp: penalty_scale (per-coefficient penalty multipliers) and min_peaks (minimum matched peaks required to fit the polynomial). Its penalty argument is set by regularization rather than through
n.traces	The n.traces argument is deprecated. Please use n_traces instead.
n.zeros	The n.zeros argument is deprecated. Please use n_zeros instead.
maxshift	Deprecated and unused. See max_shift.

Details

With alg = "lptw", peaks are found in each group average by [get_peaks](#) at its default settings and matched to the peaks of the reference group's average, so landmarks come from the averaged signal rather than from individual samples.

Value

If what = "corrected.values", a list of warped chromatogram matrices in the same order as chrom_list, with each sample warped using the warp coefficients estimated from its group average. If what = "models", a ptw_list (alg = "ptw") or lptw_list (alg = "lptw") with one model per sample, in the same order.

Author(s)

Ethan Bass

See Also

[correct_rt](#), [ptw](#)

Examples

```
data(Sa_pr)
groups <- c("a", "a", "b", "b")
warp <- correct_rt_group(chrom_list = Sa_pr, lambdas = 210, groups = groups)
```

crop_chroms	<i>Crop chromatograms</i>
-------------	---------------------------

Description

Restricts chromatograms to a range of retention times and/or wavelengths by subsetting the existing grid.

Usage

```
crop_chroms(x, rt_range = NULL, lambda_range = NULL)
```

Arguments

x	A numeric matrix (time $\times$ wavelength) or a list of such matrices. Row names must correspond to time points and column names to wavelengths.
rt_range	Numeric vector of length 2 giving the inclusive range of retention times to keep, in the time unit of the data. Either element may be NA for an open bound. Defaults to NULL, which keeps every time point.
lambda_range	Numeric vector of length 2 giving the inclusive range of wavelengths to keep. Either element may be NA for an open bound. Defaults to NULL, which keeps every wavelength.

Details

Cropping is pure subsetting: the returned data are a submatrix of the input, so values and dimnames pass through verbatim. This is the difference from cropping via [preprocess](#), which reaches a narrower range by interpolating onto a new axis — resampling the signal unless the requested grid happens to coincide with the existing time points, and reformatting the retention times either way.

Bounds are inclusive, and either end of a range may be NA to leave it open. Ranges are expressed in the units of the data itself: retention times in whatever unit the chromatogram carries, as returned by [get_times](#), and wavelengths as labeled in the column names, as returned by [get_lambdas](#).

An "estimated_baseline" attribute attached by [preprocess](#) or [correct_baseline](#) is cropped alongside the data, so it stays aligned and can still be retrieved with [get_baseline](#).

A range that does not overlap the data at all is an error. A range that merely extends past one end is honored as far as the data allow, with a warning reporting what was actually kept, since this is usually a typo or a unit mismatch. Cropping chromatograms whose time axes are offset from one another — as unaligned chromatograms generally are — can leave them with different numbers of rows, which also warns: functions that read the time axis from the first chromatogram would then mislabel the rest, so align before cropping if the result needs a common time axis.

Value

The cropped data, in the same shape as x.

Author(s)

Ethan Bass

See Also[preprocess](#), [correct_baseline](#), [get_times](#), [get_lambdas](#)Other preprocessing functions: [correct_baseline\(\)](#), [preprocess\(\)](#), [subtract_blanks\(\)](#)**Examples**

```
data(Sa_pr)
cropped <- crop_chroms(Sa_pr, rt_range = c(12, 16))
trimmed <- crop_chroms(Sa_pr, lambda_range = c(220, NA))
```

filter_peaks

*Filter peak lists***Description**

Filters peaks from a `peak_list` object by height, area, width, retention time, or fit diagnostics (converged and window_limited).

Usage

```
filter_peaks(
  peak_list,
  min_height,
  min_area,
  min_sd,
  max_sd,
  min_rt,
  max_rt,
  min_fwhm,
  max_fwhm,
  converged,
  window_limited
)
```

Arguments

<code>peak_list</code>	A <code>peak_list</code> object, consisting of a nested list of peak tables, where the first level is the sample, and the second level is the spectral wavelength. Every sample $\times$ wavelength component is described by a <code>data.frame</code> where every row represents a peak, and the columns contain information on retention time, peak shape, height, and area.
<code>min_height</code>	Minimum peak height. Peaks whose height is NA, such as failed fits kept by get_peaks , are dropped if <code>min_height</code> is supplied and retained otherwise.

min_area	Minimum peak area. Peaks whose area is NA are treated the same way.
min_sd	The min_sd argument is deprecated. Please use min_fwhm instead.
max_sd	The max_sd argument is deprecated. Please use max_fwhm instead.
min_rt	Minimum retention time.
max_rt	Maximum retention time.
min_fwhm	Minimum peak width, as a full width at half maximum, in the time units of the peak list. Peaks whose FWHM could not be measured (NA) are retained.
max_fwhm	Maximum peak width, as a full width at half maximum, in the time units of the peak list. Peaks whose FWHM could not be measured (NA) are retained.
converged	Logical. If TRUE, retains only peaks whose fit converged; if FALSE, retains only those that did not. A peak has converged == 0 either because the optimizer stopped short of convergence or because no optimizer ran at all, the latter occurring when the integration window held fewer points than the model has parameters. Requires a converged column, which peak lists made with fit = "raw" or before chromatographR 0.9.0 lack. Peaks whose converged is NA are retained.
window_limited	Logical. If FALSE, drops peaks whose fitted shape hit a limit set by the integration window or the sampling interval, meaning the window or the sampling rather than the data determined the reported shape; if TRUE, retains only those. Requires a window_limited column, which peak lists made with fit = "raw" or before chromatographR 0.9.0 lack. Peaks whose window_limited is NA are retained.

Value

A filtered peak_list object containing only peaks that satisfy the specified criteria. The call is appended to the filters attribute, as a line of code with numbers to 15 significant digits; print shows it, and [get_peaktable](#) copies it into args\$filters.

Note

The structure of filter_peaks is adapted from the filterPeaks function in Ron Wehrens' [alsace](#) package.

Author(s)

Ron Wehrens, Ethan Bass

See Also

[get_peaks](#), [filter_peaktable](#)

Examples

```
data(Sa_warp)
pks <- get_peaks(Sa_warp[1], lambda = "210")
filter_peaks(pks, min_height = 100)
filter_peaks(pks, converged = TRUE, window_limited = FALSE)
```

filter_peaktable	<i>Filter peak table</i>
------------------	--------------------------

Description

Filters features (columns) in a `peak_table`. Filtering is applied consistently across all components of the peak table, including the intensity matrix (`tab`), feature metadata (`pk_meta`), and reference spectra (if present). Filtering can be based on retention time, wavelength, feature intensity summarized across samples (mean, median, or max), feature sparsity (i.e., proportion of zero values across samples), and/or fit convergence.

Usage

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

Arguments

<code>peak_table</code>	A <code>peak_table</code> object from get_peaktable .
<code>rts</code>	Vector of retention times used to select features. Values are mapped to the closest retention times in the peak table. If no match is found within <code>tol</code> , the value is ignored and a warning is issued. Cannot be combined with <code>min_rt</code> or <code>max_rt</code> .
<code>min_rt</code>	Minimum retention time for features to be retained.
<code>max_rt</code>	Maximum retention time for features to be retained.
<code>min_value</code>	Minimum threshold for feature intensity summarized across samples (using the method specified by <code>what</code>).
<code>max_zeros</code>	Maximum allowed feature sparsity (proportion of zero values across samples).
<code>lambda</code>	Wavelength(s) used to select features in the peak table. Only features matching the specified wavelength(s) are retained.
<code>min_converged</code>	Minimum value of the converged row of <code>pk_meta</code> , between 0 and 1. That row summarizes converged over the peaks in each feature by the <code>summarize_by</code> method given to get_peaktable : with "mean", it is the proportion of those peaks whose fit converged, so 0.8 retains features that converged in at least 80% of them; with the default "weighted.mean", that proportion is weighted by peak intensity. Requires a converged row in <code>pk_meta</code> , so it is unavailable for peak tables built from <code>fit = "raw"</code> peak lists.

what	Method used to summarize feature intensity across samples for filtering. One of "mean", "median" (default), or "max".
tol	Tolerance for matching of retention times to rts. A feature is considered a match if the absolute difference is at most tol. Defaults to 0, which requires an exact match.

Value

A filtered `peak_table` containing only features (columns) that satisfy the specified criteria. The same feature indices are applied to all `peak_table` components. The call is appended to `args$filters`, as a line of code with numbers to 15 significant digits.

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)
```

get_baseline	<i>Retrieve estimated baseline</i>
--------------	------------------------------------

Description

Retrieves the baseline retained by [preprocess](#) or [correct_baseline](#) when called with `attach_baseline = TRUE`. This function does not estimate a baseline itself.

Usage

```
get_baseline(x)
```

Arguments

x	A chromatogram, or a list of chromatograms, returned by preprocess or correct_baseline with <code>attach_baseline = TRUE</code> .
---	---

Details

The baseline is dropped by [correct_rt](#), which warps the signal along the time axis, and by [subtract_blanks](#), after which it no longer describes the signal. Retrieve it before either step. [crop_chroms](#) keeps it, cropped to match.

Value

A matrix of the same dimensions as `x` containing the estimated baseline, or a list of such matrices if `x` is a list.

See Also

Other accessors: [get_lambdas\(\)](#), [get_time_resolution\(\)](#), [get_times\(\)](#)

Examples

```
data(Sa)
pr <- preprocess(Sa[[1]], dim1 = seq(10, 18.66, by = .01),
                 dim2 = seq(200, 318, by = 2), attach_baseline = TRUE)
bl <- get_baseline(pr)
```

`get_lambdas`*Get lambdas*

Description

Get wavelengths from a chromatogram, a list of chromatograms, or a `peak_table` object.

Usage

```
get_lambdas(x)
```

Arguments

`x` A chromatogram, a list of chromatograms, or a `peak_table` object.

Value

A numeric vector of wavelengths, taken from the first chromatogram if `x` is a list.

See Also

Other accessors: [get_baseline\(\)](#), [get_time_resolution\(\)](#), [get_times\(\)](#)

Examples

```
data(Sa_warp)
get_lambdas(Sa_warp)
```

get_peaks

*Get peak list***Description**

Detects chromatographic peaks, fits peak models, and extracts peak parameters at one or more wavelengths.

Usage

```
get_peaks(
  chrom_list,
  lambdas,
  fit = c("egh", "bemg", "gaussian", "raw"),
  windows = NULL,
  targets = NULL,
  tol = 0.1,
  select_by = c("height", "proximity", "none"),
  mod_list = NULL,
  sd_max = NULL,
  max_iter = 1000,
  time_unit = NULL,
  baseline = c("none", "flat", "sloped"),
  estimate_purity = FALSE,
  purity_args = list(),
  show_progress = NULL,
  cl = 2,
  collapse = FALSE,
  time.units = NULL,
  sd.max = NULL,
  max.iter = NULL,
  ...
)
```

Arguments

chrom_list	A list of chromatograms in matrix format (timepoints × wavelengths).
lambdas	A character or numeric vector specifying the wavelengths to find peaks at. Omit for one-dimensional chromatograms, or when targets has a lambda column.
fit	Peak model. Options are bidirectional exponentially modified Gaussian ("bemg"), exponential-Gaussian hybrid ("egh"), "gaussian", or "raw". "raw" integrates the signal directly by trapezoidal approximation without fitting a model. Defaults to "egh", or to "raw" when windows is provided, which allows no other value. See <i>Peak models</i> .
windows	Optional two-column matrix or data.frame of start and end retention times (or a single pair of times), giving manual integration windows applied to every chromatogram. If provided, peak detection is skipped and each window is integrated

	in every sample. Requires <code>fit = "raw"</code> . Cannot be combined with <code>targets</code> . Defaults to <code>NULL</code> . See <i>Manual integration windows</i> .
<code>targets</code>	Target retention times, as a numeric vector or a table of targets (see targeted_analysis). If provided, only peaks whose apex falls within <code>tol</code> of a target are fit, and get_peaktable groups peaks by target instead of clustering them. A <code>lambda</code> column replaces <code>lambdas</code> , and a <code>tol</code> column replaces the <code>tol</code> argument, which must then be omitted. Defaults to <code>NULL</code> , which fits every detected peak.
<code>tol</code>	Numeric. The maximum distance between a peak's apex and its target in <code>targets</code> , in minutes, regardless of the time unit of <code>chrom_list</code> . Give one value for all targets, or one per target. Defaults to <code>0.1</code> . Ignored if <code>targets</code> is <code>NULL</code> .
<code>select_by</code>	Character. Which peak to keep when more than one falls in a target's window: "height" (the default) keeps the most intense, "proximity" the one closest to the target, and "none" all of them. Ignored if <code>targets</code> is <code>NULL</code> .
<code>mod_list</code>	Optional warping models from correct_rt (with <code>what = "models"</code>). If provided, the times in <code>targets</code> are read on the aligned time axis, so they can be copied from the <code>rt</code> row of an aligned peak table. Peaks are still detected and fitted on the unaligned data; use correct_peaks afterwards to add corrected retention times. Requires <code>targets</code> .
<code>sd_max</code>	Deprecated. Filter peaks by width with filter_peaks instead, using <code>min_fwhm</code> and <code>max_fwhm</code> . A standard deviation in timepoints, converted to a FWHM cutoff of $2.355 * sd_max * dt$ in the units of <code>time_unit</code> , where <code>dt</code> is the chromatogram's time step; wider peaks are discarded. <code>NULL</code> disables width filtering. Ignored if windows is provided.
<code>max_iter</code>	Maximum number of iterations for non-linear least squares in fit_peaks . Defaults to <code>1000</code> .
<code>time_unit</code>	Units for <code>sd</code> , <code>FWHM</code> , <code>tau</code> , <code>tau_right</code> , <code>tau_left</code> , and <code>area</code> . Options are minutes ("min"), seconds ("s"), or milliseconds ("ms"). Defaults to the time unit of <code>chrom_list</code> (minutes if none is recorded). <code>rt</code> , <code>start</code> and <code>end</code> are not converted: they stay in the units of the chromatogram's own time axis. ChemStation reports areas with time in seconds, so use "s" to compare with it.
<code>baseline</code>	Whether to measure each peak above a local baseline: "none" (the default), "flat" (a constant offset per peak), or "sloped" (an offset that varies linearly across the peak). See <i>Local baselines</i> .
<code>estimate_purity</code>	Logical. Whether to estimate purity. Defaults to <code>FALSE</code> . When <code>TRUE</code> , calls get_purity for every peak in every sample, at a cost that grows with the number of wavelengths.
<code>purity_args</code>	A named list of additional arguments passed to get_purity , e.g. <code>list(background_correction = "none", metric = "max")</code> . A <code>lambdas</code> element here selects the wavelengths used to compare spectra, not the wavelengths at which peaks are integrated.
<code>show_progress</code>	Logical. Whether to show progress bar. Defaults to <code>TRUE</code> if pbapply is installed.
<code>cl</code>	Number of cores (integer) or a cluster object from makeCluster . Defaults to <code>2</code> . On Windows, only a cluster object runs in parallel; an integer runs serially.
<code>collapse</code>	Logical. Whether to combine the peaks from every wavelength into a single <code>data.frame</code> per sample. Defaults to <code>FALSE</code> .

<code>time.units</code>	The <code>time.units</code> argument is deprecated. Please use <code>time_unit</code> instead.
<code>sd.max</code>	The <code>sd.max</code> argument is deprecated. Please use <code>sd_max</code> instead.
<code>max.iter</code>	The <code>max.iter</code> argument is deprecated. Please use <code>max_iter</code> instead.
<code>...</code>	Additional arguments passed to <code>find_peaks</code> , such as <code>smooth_window</code> , <code>smooth_type</code> , <code>min_prominence_snr</code> , <code>min_curvature</code> and <code>min_amplitude</code> . See <i>Choosing the smoothing window</i> in <code>find_peaks</code> .

Details

Peaks are detected by `find_peaks` as zero-crossings in the smoothed first derivative of each trace, then fitted by `fit_peaks` according to `fit` (see *Peak models* and *Fitting procedure*). Peak areas are the trapezoidal integral of the fitted curve (or, for `fit = "raw"`, of the signal) above the local baseline (see *Local baselines*). To integrate fixed retention-time ranges instead, see *Manual integration windows*.

Value

An S3 object of class `peak_list`: a nested list (sample, then wavelength) of `data.frames`, or a list of one `data.frame` per sample when `collapse = TRUE`, with one row per peak and columns:

- `rt`: The retention time of the peak maximum. For the fitted models it generally falls between timepoints, so match it to the time axis with a tolerance rather than `==` or `match()`; `get_spectra` with `what = "rt"` returns the spectrum at the nearest timepoint.
- `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 (σ) of the Gaussian component of the fitted peak shape. This is a parameter of the model rather than the width of the peak: for `"egh"` and `"bemg"` the asymmetry of the peak is carried by `tau` or by `tau_left/tau_right`, so σ describes only the core of the peak and is not comparable across values of `fit`. Use FWHM to compare widths. NA when `fit = "raw"`, which fits no model; the width of the integration window is available as `end - start`.
- `tau`: Empirical skewness parameter (in units of time) determining peak asymmetry when `fit == "egh"`.
- `tau_right`: Time constant (in units of time) of the exponential tail on the right side of the peak when `fit = "bemg"`. A larger value gives a longer tail, and a value near 0 means no tail on that side.
- `tau_left`: Time constant (in units of time) of the exponential tail on the left side of the peak when `fit = "bemg"`. A larger value gives a longer tail, and a value near 0 means no tail on that side.
- `h`: The amplitude parameter of the fitted BEMG shape function (only when `fit = "bemg"`). It scales the shape but is not its value at center, so use the `height` column for the height of the peak.
- `center`: The center parameter of the fitted BEMG shape function (only when `fit = "bemg"`). This is the location around which the exponential tails are applied. Because BEMG peaks can be asymmetric (`tau_right != tau_left`) and a sloped baseline shifts the fitted curve, the actual peak maximum (`rt`) may differ from `center`.

- **FWHM**: The full width at half maximum of the peak. For "gaussian" this is $2\sqrt{2\ln 2}\sigma \approx 2.355\sigma$; for "egh" it is the closed form $\sqrt{(\tau \ln 2)^2 + 8\sigma^2 \ln 2}$ implied by the model (Lan & Jorgenson, 2001); and for "bemg", which has no closed form, it is measured numerically from the fitted shape. For "raw" it is measured directly from the signal above the local baseline, and is NA when the integration window does not fall below half the height of the peak on both sides, as for a peak fused with a neighbor. Unlike sd, FWHM is comparable across values of fit.
- **height**: The height of the peak above the fitted baseline.
- **area**: The area of the peak above the fitted baseline, estimated by trapezoidal approximation.
- **r.squared**: Coefficient of determination (R^2) of the fitted curve over the peak window, $1 - \text{SS}_{\text{res}} / \text{SS}_{\text{tot}}$. Negative when the fit is worse than a flat line at the mean. Not returned when fit = "raw".
- **purity**: The peak purity as estimated by `get_purity`. NA unless estimate_purity = TRUE.
- **window_limited**: 1 if a fitted shape parameter hit a limit set by the integration window, or a "gaussian" or "egh" fit narrowed to one sampling interval at half height, meaning the window or the sampling rather than the data determined the reported shape, and 0 otherwise. Not returned when fit = "raw".
- **converged**: 1 if the nonlinear least squares optimizer reported convergence for the final stage of the fit, and 0 otherwise. A 0 means the reported parameters are wherever the optimizer stopped rather than a converged fit, so treat the corresponding height and area with caution. It also covers the case where part or all of the fit was skipped because the integration window holds too few points to determine it; the peak then keeps the last step that ran, or its starting values (see *Fitting procedure*). `get_peaks` warns when more than 20% of peaks sit in such a window. Not returned when fit = "raw".
- **floor**: Constant baseline level (only when baseline = "flat").
- **floor_start**: Baseline level at the left edge of the peak window (only when baseline = "sloped").
- **floor_end**: Baseline level at the right edge of the peak window (only when baseline = "sloped"). For the peak models, floor, floor_start and floor_end are constrained to be non-negative.
- **target**: The row of targets the peak was selected for (only when targets is supplied).

Peak models

The three peak models are fitted to each peak by nonlinear least squares; "raw" uses no model.

- **"gaussian"**: a symmetric Gaussian, with a center, a width (sd) and a height.
- **"egh"**: the exponential-Gaussian hybrid of Lan & Jorgenson (2001), a Gaussian whose width grows on one side of the apex. It adds one parameter, tau, which is positive for a tailing peak and negative for a fronting one.
- **"bemg"**: the average of two exponentially modified Gaussians, one with an exponential tail to the left and one to the right. Their time constants, tau_left and tau_right, set the two tails independently, so a peak can tail, front, or both; a value near 0 means no tail on that side.
- **"raw"**: no model. The signal itself is integrated across the window, so r.squared, window_limited and converged are not reported.

"gaussian" cannot follow tailing or fronting; "egh" follows one or the other, but not both in the same peak; "bemg" follows both. The more flexible models have more parameters (3, 4 and 5 respectively for the peak shape, plus one more for a "flat" baseline or two more for a "sloped" baseline).

Local baselines

A peak's height and area are measured over its integration window, which runs from the valley before the peak to the valley after it, as found by [find_peaks](#), or spans the manual window if the windows argument is given.

- With baseline = "none", height and area are measured from zero, so the chromatogram should already be baseline corrected (see [preprocess](#)).
- With the fitted peak models ("gaussian", "egh", "bemg"), a "flat" or "sloped" baseline is fitted together with the peak shape by nonlinear least squares. It cannot be negative; a flat baseline is capped at the lowest intensity in the window, and each end of a sloped one is capped at the median intensity of the first or last 2% of the window. height and area are those of the fitted shape above the baseline, with the area integrated across the integration window only.
- With fit = "raw", nothing is fitted: a "flat" baseline is a horizontal line at the lower of the two boundary intensities, and a "sloped" one a straight line between them. Where that line lies above the signal, it subtracts area, so the area can come out negative; [get_peaktable](#) records a negative area as 0.

With "flat" or "sloped", each peak gets its own baseline. This differs from drop-line integration in vendor software such as ChemStation, where a vertical line from each valley meets a baseline shared by the whole cluster of peaks.

Fitting procedure

A fit is built up in successive steps: first the peak shape, then a constant baseline, and finally its slope. To fit the full model, a peak's integration window must have at least as many timepoints as there are parameters in the model. A step is skipped if the window has too few timepoints to fit all the required parameters, and discarded if the optimizer cannot start from its initial values or, for "bemg", if a fitted tail rises towards the edge of the window instead of falling back towards the baseline. Either way, the peak keeps the last step that worked. A "sloped" baseline can therefore come out flat, or a baseline can be left at its starting estimate rather than fitted, and if even the shape fails, every parameter will keep its starting value. In each of these cases, converged is set to 0.

Peaks that can't be fitted at all are dropped, unless they come from targets. Those are kept, with height and area set to NA and converged to 0, so that every target found has a row.

Manual integration windows

Manual windows integrate every sample over the same retention-time ranges, so each compound is measured the same way in every sample, and every sample gets the same set of peak table columns. The raw signal in each window is integrated whether or not a peak is detected there, so a window with no peak gets an area near zero. With baseline = "none", two adjacent windows sharing a boundary time act as a vertical drop line: their areas sum exactly to the area of their union, so a peak can be split at a chosen time. The "flat" and "sloped" baselines are drawn within each window separately, so their areas do not sum to that of the union. To fit peak models at known retention times, use targets instead.

Note

Adapted from getAllPeaks by Ron Wehrens (<https://github.com/rwehrens/alsace/blob/master/R/getAllPeaks.R>); however, the underlying algorithms for finding and integrating peaks differ.

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.
- Arase, Shuntaro, Kanta Horie, Takashi Kato, et al. 2016. Intelligent Peak Deconvolution through In-Depth Study of the Data Matrix from Liquid Chromatography Coupled with a Photo-Diode Array Detector Applied to Pharmaceutical Analysis. *Journal of Chromatography A*, 1469: 35–47. doi:10.1016/j.chroma.2016.09.037.
- 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 multi-variate curve resolution: the alsace package for R. *Metabolomics* 11:143-154. doi:10.1007/s1130601406835.

See Also

[find_peaks](#), [fit_peaks](#), [print_peak_list](#), [.peak_list](#), [targeted_analysis](#)

Examples

```
data(Sa_pr)
pks <- get_peaks(Sa_pr, lambdas = c('210'), fit = "egh")
pks <- filter_peaks(pks, max_fwhm = 1)
```

get_peaktable

Convert peak lists into an aligned peak table

Description

Converts a peak_list object containing peaks from multiple samples into an aligned peak table by clustering peaks across samples according to retention time and (optionally) spectral similarity. The tree is cut at height hmax, whose meaning depends on clust (see hmax).

Usage

```
get_peaktable(peak_list, ...)

## Default S3 method:
get_peaktable(
  peak_list,
  chrom_list,
  response = c("area", "height"),
  use_corrected_rt = NULL,
  hmax = 0.2,
  summarize_by = c("weighted.mean", "median", "mean", "max"),
  plot_it = FALSE,
  ask = plot_it,
  clust = c("rt", "sp.rt"),
  sigma_t = NULL,
  sigma_r = 0.5,
  deepSplit = FALSE,
  verbose = FALSE,
  ...,
  sigma.t = NULL,
  sigma.r = NULL,
  use.cor = NULL
)

## S3 method for class 'matched_peak_list'
get_peaktable(
  peak_list,
  pks,
  response = c("area", "height"),
  rt_tol = 0.01,
  na.fill = 0,
  verbose = FALSE,
  ...
)

## S3 method for class 'matched_peak_list_set'
get_peaktable(
  peak_list,
  pks,
  response = c("area", "height"),
  rt_tol = 0.01,
  na.fill = 0,
  verbose = FALSE,
  ...
)
```

Arguments

peak_list	A peak_list object created by get_peaks , or a matched_peak_list created by match_peaks .
...	Additional arguments (currently unused).
chrom_list	A list of chromatographic matrices. Only required when clust = "sp.rt".
response	Peak intensity measure: either "area" (default) or "height".
use_corrected_rt	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. Either way, they are stored in the rt, start and end rows of pk_meta, and args\$use_corrected_rt records which were used.
hmax	Height at which the clustering tree is cut. With clust = "rt", peaks are clustered by complete linkage on retention time, so hmax is the largest retention time difference allowed between any two peaks in the same cluster, in minutes. With clust = "sp.rt", peaks are clustered by average linkage on the dissimilarity $1 - S_{ij}$, which runs from 0 to 1, and hmax is the maximum tree height passed to cutreeDynamicTree ; clusters may be split below it (see deepSplit). There, the default of 0.2 lets peaks with identical spectra sit about $0.67 * \text{sigma_t}$ apart, a third of the mean end - start at the default sigma_t. Ignored for peak lists produced by get_peaks (windows =) or get_peaks (targets =), since those peaks are grouped by integration window or target instead of being clustered (see <i>Windows and targets</i>).
summarize_by	How to summarize peak metadata within each cluster. Options are "weighted.mean" (default, weights peaks by their intensity as given by response), "mean", "median" (which aggregate metadata across all peaks in the cluster) or "max" (which selects the most intense peak in the cluster and uses its metadata directly).
plot_it	Logical. If TRUE, for every component a strip plot will be shown indicating the clustering.
ask	Logical. Ask before showing new plot? Defaults to plot_it.
clust	Specify whether to perform hierarchical clustering based on spectral similarity and retention time (sp.rt) or retention time alone (rt). Defaults to rt. Ignored for peak lists produced by get_peaks (windows =) or get_peaks (targets =), which are grouped by window or target (see <i>Windows and targets</i>).
sigma_t	Width of Gaussian kernel for retention time similarity, in minutes. Controls the weight given to retention time when clust = "sp.rt". Defaults to half the mean end - start across all peaks.
sigma_r	Width of Gaussian kernel for spectral similarity. Controls the weight given to spectral correlation when clust = "sp.rt". Defaults to 0.5.
deepSplit	Logical. Controls sensitivity to cluster splitting when clust = "sp.rt". If TRUE, returns a larger number of smaller clusters. Defaults to FALSE. For additional information, see cutreeDynamicTree .
verbose	Logical. Whether to warn when two peaks from the same sample fall in the same cluster. Defaults to FALSE.

<code>sigma.t</code>	The <code>sigma.t</code> argument is deprecated. Please use <code>sigma_t</code> , which is in minutes. <code>sigma.t</code> is read in the units of the retention times, as before.
<code>sigma.r</code>	The <code>sigma.r</code> argument is deprecated. Please use <code>sigma_r</code> instead.
<code>use.cor</code>	The <code>use.cor</code> argument is deprecated. Please use <code>use_corrected_rt</code> instead.
<code>pks</code>	A <code>peak_list</code> object from get_peaks . Required when <code>peak_list</code> is a <code>matched_peak_list</code> . Used to retrieve per-peak intensities and metadata.
<code>rt_tol</code>	Numeric. Maximum allowed deviation (in retention-time units) when joining matched peak RTs back to the corresponding row in <code>pks</code> . Defaults to 0.01.
<code>na.fill</code>	Value used to fill cells where no peak was matched in a given sample. Defaults to 0. Only used by the <code>matched_peak_list</code> method.

Value

An S3 [peak_table](#) object, containing the following elements:

- `tab`: The peak table itself — a `data.frame` of intensities in a sample $\times$ peak configuration. A peak that was not found in a sample, or whose response is negative, is recorded as 0 (`na.fill` for a `matched_peak_list`). A peak whose model failed, which `get_peaks(targets = )` keeps with a missing height and area, is recorded as NA.
- `pk_meta`: A `data.frame` of peak metadata, one column per peak. Its rows are the wavelength (`lambda`), the peak number (`peak`), and whichever of the peak list's `rt`, `start`, `end`, `sd`, `tau`, `FWHM`, `r.squared`, `purity`, `window_limited` and `converged` columns it has. See the Value section of [get_peaks](#) for their meanings. The rows taken from the peak list are summarized across samples according to `summarize_by`; by default, peaks are weighted by intensity, which reduces the influence of small peaks. With "weighted.mean" or "mean", the `converged` and `window_limited` rows become the share of the column's peaks with that flag, so 0.25 means a quarter of them. In a table built from a `matched_peak_list`, the rows hold the reference sample's values instead.
- `sample_meta`: A `data.frame` of sample metadata: the fields added to the chromatograms with `chromConverter::add_metadata()` (`chromConverter` 0.11.0 or later), and any attached with [attach_metadata](#). NA if there are none.
- `instrument_meta`: A `data.frame` of instrumental metadata transferred automatically from the chromatograms.
- `ref_spectra`: A matrix of reference spectra (in a wavelength $\times$ peak configuration). NA until added with [attach_ref_spectra](#).
- `args`: A list of the arguments given to [get_peaktable](#) to generate the peak table, with the `fit`, `baseline` and `max_iter` the peaks were fitted with in [get_peaks](#). Its `filters` element holds any [filter_peaks](#) and [filter_peaktable](#) calls applied, as code, separated by "; ".

Clustering

By default, clustering is performed on retention times alone (when `clust = "rt"`). Alternatively, clustering can also incorporate information about spectral similarity (when `clust = "sp.rt"`) using a similarity function adapted from Broeckling et al., 2014:

$$S_{ij} = \exp\left(-\frac{(t_i - t_j)^2}{2\sigma_t^2}\right) \cdot \exp\left(-\frac{(1 - c_{ij})^2}{2\sigma_r^2}\right)$$

where c_{ij} is the spectral correlation between spectra i and j , and σ_t and σ_r control the relative contributions of retention time and spectral similarity, respectively (see `sigma_t` and `sigma_r`).

When two peaks from the same sample fall in the same cluster, only the more intense one enters the peak table, with a warning if `verbose = TRUE`. Such warnings are often benign, but they can also mean `hmax` is too large. Lowering `hmax` merges fewer peaks, at the cost of splitting some across multiple clusters; filtering low-intensity peaks before clustering reduces the warnings without that trade-off.

Windows and targets

Peak lists produced by `get_peaks(windows = )` are grouped by integration window rather than clustered by retention time, including after `correct_peaks`. This gives one peak table column per window, ordered by retention time.

Peak lists produced by `get_peaks(targets = )` are grouped by target instead of being clustered, each peak going to the matching target. Targets found in no sample are omitted from the peak table. If `targets` has a name column, the columns are named after the targets.

Peak tables built from `get_peaks(windows = )` or `get_peaks(targets = )` keep those windows or targets as an attribute of the same name, as a record of how the table was built. The `targets` attribute lists every target that was searched for, including any found in no sample, which have no column. Neither attribute is updated when columns are removed.

Matched peak lists

`match_peaks` pairs the peaks in each sample with the peaks of a reference sample, by spectrum as well as retention time. Each column then follows one compound across the samples, even where retention times drift too far for clustering to group the peaks reliably. In crowded stretches of the chromatogram, this works only if neighboring compounds have distinct spectra. This suits targeted analysis (see `targeted_analysis`) and studies in which one sample, such as a standard mix or a pooled sample, contains every compound of interest.

The peak table has one column for each peak in the reference sample, at each of the specified wavelengths. A peak that's missing in the reference sample gets no column, even if it's present in other samples, so choose the reference with care (see `match_peaks`).

Note

This function is adapted from the `getPeakTable` function in the `alsace` package by Ron Wehrens: <https://github.com/rwehrens/alsace/blob/master/R/getPeakTable.R>.

Author(s)

Ron Wehrens, Ethan Bass

References

- Broeckling, C. D., Afsar F.A., Neumann S., Ben-Hur A., and Prenni J.E. 2014. RAM-Clust: 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 multi-variate curve resolution: the alsace package for R. *Metabolomics* 11:143-154. doi:10.1007/s1130601406835.

See Also

[peak_table](#), [attach_ref_spectra](#), [attach_metadata](#)

Examples

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

get_spectra	<i>Extract a spectrum from a chromatogram</i>
-------------	---

Description

Extracts the UV-VIS spectrum at a specified retention time from one or more chromatograms in a `chrom_list`.

Usage

```
get_spectra(
  loc,
  peak_table,
  chrom_list = NULL,
  idx = seq_along(chrom_list),
  scale_spectrum = FALSE,
  what = c("peak", "rt", "idx"),
  format = c("wide", "long")
)
```

Arguments

<code>loc</code>	Peak or retention time to extract the spectrum at. Interpretation depends on <code>what</code> .
<code>peak_table</code>	A <code>peak_table</code> object. Required if <code>what == "peak"</code> .
<code>chrom_list</code>	A list of chromatographic matrices. If <code>NULL</code> , the list is found through the pointer in <code>peak_table</code> .
<code>idx</code>	Indices of chromatograms to extract spectra from. Defaults to all chromatograms in <code>chrom_list</code> .
<code>scale_spectrum</code>	Logical. Whether to normalize absorbance values to the range [0, 1]. Defaults to <code>FALSE</code> .

what	How to interpret loc. One of "peak" (look up retention time from peak_table), "rt" (treat loc as a retention time directly), or "idx" (treat loc as a row index). Defaults to "peak".
format	One of "wide" or "long". In wide format, wavelengths are rows and each column is a sample. In long format, columns are wavelength, absorbance, sample, and rt. Defaults to "wide".

Value

Returns a `data.frame` in the format specified by `format`.

Examples

```
data(pk_tab)
data(Sa_warp)
get_spectra("V8", peak_table = pk_tab)
```

get_time_resolution	<i>Get time resolution</i>
---------------------	----------------------------

Description

Get the sampling interval of a chromatogram: the median gap between consecutive retention times.

Usage

```
get_time_resolution(x, idx = 1, digits = 6, unit = NULL)
```

Arguments

x	A chromatogram, a list of chromatograms, or a <code>peak_table</code> object.
idx	Index of the chromatogram from which to derive the interval, if x is a list or <code>peak_table</code> . Defaults to 1.
digits	Number of significant digits to round to. Defaults to 6, because a regular time axis carries floating-point noise in the last digits of <code>diff</code> .
unit	Unit for the returned interval: "min", "s" or "ms". Defaults to NULL, giving the interval in the units of the chromatogram's own time axis, which is what <code>preprocess</code> 's <code>dim1</code> expects. Supply "min" to pair with the <code>time_resolution</code> arguments of <code>plot_chroms</code> and <code>reshape_chroms</code> , which are in minutes, regardless of the units of x. The units of x are read from its <code>time_unit</code> attribute, falling back to minutes when it is absent, as <code>plot_chroms</code> does.

Value

A length-one numeric giving the time between consecutive timepoints, in unit if supplied and otherwise in the units of the chromatogram's time axis.

See Also

Other accessors: [get_baseline\(\)](#), [get_lambdas\(\)](#), [get_times\(\)](#)

Examples

```
data(Sa_warp)
get_time_resolution(Sa_warp)
```

get_times

Get retention times

Description

Get retention times from a chromatogram, a list of chromatograms, or a `peak_table` object.

Usage

```
get_times(x, idx = 1)
```

Arguments

x	A chromatogram, a list of chromatograms, or a <code>peak_table</code> object.
idx	Index of the chromatogram from which to extract times, if x is a list or <code>peak_table</code> . Defaults to 1.

Value

Numeric vector of retention times from the chromatogram specified by `idx`.

See Also

Other accessors: [get_baseline\(\)](#), [get_lambdas\(\)](#), [get_time_resolution\(\)](#)

Examples

```
data(Sa_warp)
get_times(Sa_warp)
```

match_peaks

*Match peaks across chromatograms***Description**

Matches peaks across chromatograms by spectral correlation and retention time proximity. Each reference peak is matched to at most one sample peak, taking candidate pairs in order of score, highest first. The resulting `matched_peak_list` object can be passed to `correct_rt` to perform landmark parametric time warping (`alg = "lptw"`).

Usage

```
match_peaks(
  peak_list,
  chrom_list = NULL,
  reference = NULL,
  targets = NULL,
  lambda = NULL,
  rt_window = 0.3,
  spectral_cutoff = 0.95,
  background_correction = c("none", "linear"),
  rt_weight = 0.15,
  use_corrected_rt = NULL,
  spectral_test = c("cutoff", "noise"),
  alpha = 0.01,
  plot_it = FALSE,
  verbose = getOption("verbose")
)
```

Arguments

peak_list	A peak list object from get_peaks .
chrom_list	List of chromatograms in matrix format. If NULL, will be retrieved from the peak_list object.
reference	Index or name of the reference sample, or "best" to select the reference algorithmically using bestref . Defaults to NULL, which uses the first sample, or, if targets is given, a sample chosen from the targets.
targets	Optional table of targets, or a numeric vector of their retention times (see targeted_analysis). Used to choose the reference sample, unless reference is given, and to name the targets' columns in the peak table. See <i>Targets and the reference sample</i> . Cannot be combined with reference = "best". Defaults to NULL.
lambda	A character or numeric value specifying the wavelength to use for peak matching. Required if peaks were detected at multiple wavelengths, unless targets has a lambda column, which sets the wavelength per target; lambda then fills in targets without one.

rt_window	Numeric. Maximum allowed difference in retention time, in minutes, between a reference and sample peak to be considered a candidate match. An <code>rt_window</code> column in <code>targets</code> overrides it for those targets. Defaults to 0.3.
spectral_cutoff	Numeric. Minimum spectral correlation for a candidate peak pair to be retained. A <code>spectral_cutoff</code> column in <code>targets</code> overrides it for those targets. Ignored, with a warning, for one-dimensional chromatograms. Defaults to 0.95.
background_correction	How to correct the apex spectrum of each peak before correlating: "none" (the default) uses it as is, and "linear" subtracts the background under the peak, interpolated between the spectra at its start and end. See <i>Background correction</i> .
rt_weight	Numeric in [0, 1]. Penalizes retention time offset when ranking candidate pairs: each pair's spectral correlation is multiplied by $1 - \text{rt_weight} * d / \text{rt_window}$, where d is the pair's retention time difference. At <code>rt_weight</code> = 1, a pair at the edge of the window scores zero. Defaults to 0.15.
use_corrected_rt	Logical. Whether to compare corrected retention times (<code>rt.cor</code>) from correct_peaks rather than raw ones (<code>rt</code>), both between peaks and against targets. Defaults to NULL, which uses corrected times when every sample has them.
spectral_test	How a candidate pair's spectra are compared: "cutoff" (the default) requires their correlation to be at least <code>spectral_cutoff</code> ; "noise" tests whether they could be the same spectrum given each sample's noise, so that a small, noisy peak is not rejected for its noise alone. The "noise" test is experimental; see <i>Testing spectra against noise</i> .
alpha	Significance level for <code>spectral_test</code> = "noise". A pair is rejected if its p-value is below alpha. Pairs whose spectra are the same apart from noise are rejected at most about alpha of the time, except when the reference peak is small (up to roughly 20 times the noise), where they are rejected more often. Pairs whose spectra truly correlate at <code>spectral_cutoff</code> are rejected more often too, up to about half the time for peaks far above the noise, as with a correlation threshold. Defaults to 0.01.
plot_it	Logical. Whether to plot the matches with plot.matched_peak_list . Defaults to FALSE.
verbose	Whether to print messages to console.

Value

A `matched_peak_list` object: a list with one element per sample containing the matched peak pairs, per-pair spectral correlation scores, and peak data in index space for use with [correct_rt](#). A sample with no peaks or no matches is NULL, with a warning. When `targets` is given, the targets with `rt` replaced by the reference's own retention times are stored as the "reference_targets" attribute. When the targets span several wavelengths, peaks are matched separately at each, and the result is a `matched_peak_list_set`: a list of `matched_peak_list` objects named by wavelength, which [get_peaktable](#) combines into one table. Unless reference is given, every wavelength uses the same reference sample, chosen from all the targets; if no sample contains every target, each wavelength gets its own, with a warning.

Targets and the reference sample

With targets, a sample's peak for a target is its tallest peak within `tol` of it that, when fitted, has `r_squared` of at least 0.5; a peak within `tol` of several targets counts only for the nearest. Unless reference is given, the reference is chosen among samples with a peak for every target, where a peak counts if it is at least a tenth of that target's median height across samples. The sample whose weakest target is largest relative to its median is chosen, then the one whose farthest peak lies closest to its target time.

Background correction

With `background_correction = "linear"`, the apex spectrum is averaged over 2 scans either side and each boundary spectrum over 5 scans either side, to limit the noise the subtraction adds. The subtraction removes a background that varies linearly across the peak, such as a sloping baseline or a neighbor's tail. It also adds noise, and where a boundary lies on a neighboring peak it subtracts part of that peak's spectrum.

Testing spectra against noise

With `spectral_test = "noise"`, each candidate pair is tested rather than thresholded. The sample spectrum is fitted as an offset plus a multiple of the reference spectrum by least squares, each wavelength weighted by the reciprocal of the sample's noise there. The hypothesis is that, without noise, the two spectra would correlate at `spectral_cutoff`; the residual that this mismatch would leave is subtracted, and what remains is compared with the residuals left under the same fit by pairs of noise spectra, one from the sample and one from the reference scaled by the fitted multiple. For a peak far above the noise, this reduces to requiring a noise-weighted correlation of at least `spectral_cutoff`; for a small peak, it allows for the noise, so a real peak is not rejected because its spectrum is noisy. Each matched sample's `spectral_p` holds the p-values of its matched pairs.

This test is experimental. It has had little testing on real data, and the way its p-values are calculated may change in future versions.

The noise spectra come from each chromatogram's own peak-free timepoints: those outside every peak in `peak_list` taller than 5 times its trace's noise, at any wavelength in `peak_list`, after dropping timepoints whose spectra still stand out from the rest. A trace's noise is the scaled median absolute deviation of its timepoints outside every peak taller than 3 times the noise, found by iterating from the estimate given by its point-to-point differences, $\text{mad}(\text{diff}(y)) / \sqrt{2}$ (von Neumann 1941; Donoho and Johnstone 1994).

The test cannot tell compounds apart near the noise level, where every spectrum is consistent with every other; decide whether a small peak is present by its height instead. A peak overlapped by a neighbor or sitting on structured background may be rejected because its apex spectrum really is a mixture. The test needs `background_correction = "none"` and, in each chromatogram, at least 30 peak-free timepoints and at least $1 / \alpha$, since with fewer it could never reject a match. On short traces with strongly autocorrelated noise, the peak-free timepoints are fewer independent spectra than their number, so a `spectral_cutoff` of 1 rejects identical spectra up to about twice as often as α .

Author(s)

Ethan Bass

References

- Donoho, D. L., and Johnstone, I. M. 1994. Ideal spatial adaptation by wavelet shrinkage. *Biometrika* 81:425-455. doi:[10.1093/biomet/81.3.425](https://doi.org/10.1093/biomet/81.3.425).
- von Neumann, J. 1941. Distribution of the ratio of the mean square successive difference to the variance. *Ann. Math. Stat.* 12:367-395. doi:[10.1214/aoms/1177731677](https://doi.org/10.1214/aoms/1177731677).

See Also

[correct_rt](#), [get_peaks](#), [targeted_analysis](#)

Examples

```
data(Sa_pr)
pks <- get_peaks(Sa_pr, lambdas = 210, fit = "egh")
match_peaks(pks, lambda = 210)
```

merge_peaks	<i>Merge split peaks</i>
-------------	--------------------------

Description

Combines split peaks into a single column of the peak table.

Usage

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

Arguments

peak_table	A peak_table object 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 add the peaks together.

Details

In each sample, method = "max" keeps the largest of the specified peaks and method = "sum" adds them. The result replaces the peak with the highest mean across samples, whose metadata is retained; the other peaks are removed.

Value

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

Author(s)

Ethan Bass

See Also

Other utility functions: [combine_peaks\(\)](#), [reshape_chroms\(\)](#), [reshape_peaktable\(\)](#)

Examples

```
data(pk_tab)
pk_tab <- merge_peaks(peak_table = pk_tab, peaks = c("V8", "V9"))
```

mirror_plot	<i>Make mirror plot from peak table</i>
-------------	---

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()).
chrom_list	A list of chromatograms in matrix format (timepoints × 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(s) at which to plot the traces. Required for multi-wavelength chromatograms; ignored for one-dimensional chromatograms.
var	Name of the column in the sample metadata that defines the two groups.
subset	Character vector of the two levels of var to compare. Required if var has more than two levels.
print_legend	Logical. Whether to print legend. Defaults to TRUE.

legend_txt	Character vector containing labels for legend.
legend_pos	Legend position, used only when <code>mirror = FALSE</code> . The mirror layout always places the legends at the top and bottom right.
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 <code>matplot</code> .

Details

Compares the chromatograms in two groups, defined by the levels of `var`. When `mirror` is TRUE both are drawn in one panel, with the second inverted below the axis; when it is FALSE they are drawn in two stacked panels.

Value

No return value, called for side effects.

Side effects

Plots are rendered to the active graphics device.

Author(s)

Ethan Bass

See Also

Other visualization functions: `boxplot.peak_table()`, `plot.chrom_list()`, `plot.matched_peak_list()`, `plot.peak_list()`, `plot.peak_table()`, `plot_all_spectra()`, `plot_chrom_contour()`, `plot_chroms()`, `plot_chroms_heatmap()`, `plot_spectrum()`, `scan_chrom()`

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	<i>Normalize peak table or chromatograms</i>
----------------	--

Description

Divides each sample in a peak table or list of chromatograms by its value in a column of the sample metadata or of the peak table. For normalization by sample metadata, the metadata must first be attached to the peak_table using [attach_metadata](#).

Usage

```
normalize_data(  
  peak_table,  
  column,  
  chrom_list = NULL,  
  what = c("peak_table", "chrom_list"),  
  by = NULL,  
  on_invalid = c("warn", "error", "silent")  
)
```

Arguments

peak_table	A peak_table object.
column	The name of the column to be used for normalization.
chrom_list	List of chromatograms for normalization. Its names must match the sample names of the peak_table, in the same order. If omitted, it is retrieved using the name stored in the peak_table.
what	Output type to return: either "peak_table" (default) or "chrom_list".
by	Whether to normalize by a column in sample metadata ("meta") or by a column in the peak table ("peak"). By default, it is inferred from where column is found, and an error is raised if it is found in both.
on_invalid	How to handle invalid normalization values (i.e. zero, negative, or NA values). One of "warn" (the default), "silent", or "error". Invalid values are replaced with NA unless "error" is chosen.

Value

Either a normalized peak_table object or a normalized chrom_list, depending on the value of what.

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)

# normalize by sample mass
pk_tab <- attach_metadata(peak_table = pk_tab, metadata = meta, column="vial")
norm <- normalize_data(pk_tab, "mass", what = "peak_table")

# normalize by internal standard
norm <- normalize_data(pk_tab, column = "V14", by = "peak")
```

peak_table-class

Peak Table Object

Description

S3 objects of class `peak_table` store chromatographic peak data along with associated metadata and reference spectra.

Components

A `peak_table` object contains the following elements:

- `tab`: The peak table itself — a `data.frame` of intensities in a sample $\times$ peak configuration.
- `pk_meta`: A `data.frame` containing peak metadata (e.g., the spectral component, peak number, and summarized retention time). See [get_peaktable](#) for its rows.
- `sample_meta`: A `data.frame` of sample metadata: the fields added to the chromatograms with `chromConverter::add_metadata()` (chromConverter 0.11.0 or later), and any attached with [attach_metadata](#). NA if there are none.
- `instrument_meta`: A `data.frame` of instrumental metadata transferred automatically from the chromatograms.
- `ref_spectra`: A matrix of reference spectra (in a wavelength $\times$ peak configuration). NA until added with [attach_ref_spectra](#).
- `args`: A list of the arguments given to [get_peaktable](#) to generate the peak table, with the peak list's fitting settings. See [get_peaktable](#).

Methods

The following methods are available:

- [print.peak_table\(\)](#)
- [dim.peak_table\(\)](#)
- [head.peak_table\(\)](#)
- [tail.peak_table\(\)](#)

- `row.names.peak_table()`
- `subset.peak_table()`
- `summary.peak_table()`
- `[.peak_table`

See Also

`get_peaktable()` for creating `peak_table` objects

Examples

```
# See get_peaktable() for creation examples
# To access peak_table components:
data(pk_tab)
pk_tab$tab          # peak table
pk_tab$pk_meta      # peak metadata
pk_tab$sample_meta  # sample metadata
```

pk_tab	<i>Goldenrod peak table</i>
--------	-----------------------------

Description

A peak table generated from the exemplary goldenrod root extracts stored in [Sa](#).

Usage

```
data(pk_tab)
```

Format

A `peak_table` object.

See Also

Other data objects: [Sa](#), [Sa_pr](#), [Sa_warp](#)

plot.chrom_list	<i>Plot a list of chromatograms</i>
-----------------	-------------------------------------

Description

Plots a chrom_list, as returned by `chromConverter::read_chroms()`, as traces (`plot_chroms()`), a heatmap of samples (`plot_chroms_heatmap()`), or a contour plot of one sample (`plot_chrom_contour()`). If each sample holds several elements, the "dad" element is plotted, or for type = "traces" the one named by what.

Usage

```
## S3 method for class 'chrom_list'
plot(x, type = c("traces", "heatmap", "contour"), ...)
```

Arguments

x	A chrom_list object.
type	Type of plot: "traces" (the default), "heatmap", or "contour".
...	Arguments to the plotting function selected by type. lambdas names the traces to plot for "traces", and a single wavelength for "heatmap"; idx selects samples for "traces" and "heatmap", and one sample for "contour".

Value

The value returned by the selected plotting function.

Author(s)

Ethan Bass

See Also

Other visualization functions: `boxplot.peak_table()`, `mirror_plot()`, `plot.matched_peak_list()`, `plot.peak_list()`, `plot.peak_table()`, `plot_all_spectra()`, `plot_chrom_contour()`, `plot_chroms()`, `plot_chroms_heatmap()`, `plot_spectrum()`, `scan_chrom()`

Examples

```
data(Sa_pr)
class(Sa_pr) <- c("chrom_list", "list")
plot(Sa_pr, lambdas = 210)
plot(Sa_pr, type = "heatmap", lambdas = 210)
plot(Sa_pr, type = "contour", idx = 1)
```

`plot.matched_peak_list`*Plot matched peak pairs*

Description

Shows how peaks in each sample were matched to the reference by [match_peaks](#).

Usage

```
## S3 method for class 'matched_peak_list'
plot(x, idx = NULL, type = c("ladder", "scatter"), ...)

## S3 method for class 'matched_peak_list_set'
plot(x, ...)
```

Arguments

<code>x</code>	A <code>matched_peak_list</code> object from match_peaks , or a <code>matched_peak_list_set</code> , whose wavelengths are plotted in turn.
<code>idx</code>	Integer or character vector of sample indices or names to plot. Defaults to all non-reference samples.
<code>type</code>	Plot type, either "ladder" (default) or "scatter". See <i>Details</i> .
<code>...</code>	Additional arguments passed to plot for type = "ladder", such as <code>xlim</code> to zoom in on part of the run.

Details

With type = "ladder" (the default), each sample gets one row, with retention time on the x-axis. Dotted vertical lines mark the reference peaks, and a short tick marks each one in every row. A matched sample peak is drawn as a point filled by its spectral correlation with the reference peak, joined to the reference tick by a horizontal bar, so the length and direction of the bar show the retention time shift. A tick without a bar is a reference peak left unmatched in that sample, and an open circle is a sample peak left unmatched. Where a bar would overlap another bar or run through an unmatched peak, it is lifted slightly above its row.

With type = "scatter", each sample gets its own panel, plotting every candidate pair (a pair within the retention time window and above the spectral cutoff) at its reference and sample retention times, so retention time drift shows as distance from the $y = x$ diagonal. Candidate pairs are colored by spectral correlation, matched pairs are outlined in black, and unmatched peaks are marked on the axes.

Value

Invisibly returns `x`.

Author(s)

Ethan Bass

See Also

[match_peaks](#), [get_peaktable](#)

Other visualization functions: [boxplot.peak_table\(\)](#), [mirror_plot\(\)](#), [plot.chrom_list\(\)](#), [plot.peak_list\(\)](#), [plot.peak_table\(\)](#), [plot_all_spectra\(\)](#), [plot_chrom_contour\(\)](#), [plot_chroms\(\)](#), [plot_chroms_heatmap\(\)](#), [plot_spectrum\(\)](#), [scan_chrom\(\)](#)

plot.peak_list

Overlay fitted peak shapes on chromatograms

Description

Draws fitted peak shapes over chromatographic traces to assess integration accuracy.

Usage

```
## S3 method for class 'peak_list'
plot(
  x,
  ...,
  chrom_list,
  idx = 1,
  lambda = NULL,
  targets = NULL,
  tol = 0.1,
  pad = 0.5,
  points = FALSE,
  ticks = FALSE,
  alpha = 0.5,
  color = NULL,
  cex.points = 0.5,
  numbers = FALSE,
  cex.font = 0.5,
  digits = 2,
  y.offset = 0.03,
  plot_purity = c("none", "ratio", "curves", "curves_local"),
  purity_args = list(),
  res
)
```

Arguments

x	A peak_list object. Output from the get_peaks function.
...	Additional arguments to main plot function. The plot is titled with the sample, wavelength and peak model by default; pass main to change the title, or main = "" to omit it, and xaxt = "n" or yaxt = "n" to hide an axis.
chrom_list	List of chromatograms (retention time × wavelength matrices). If missing, extracted from environment using the pointer in x.
idx	Index or name of chromatogram to plot from chrom_list.
lambda	Wavelength (column) to use for plotting. Defaults to the first wavelength in x.
targets	Optional target retention times, as a numeric vector or a table of targets (see targeted_analysis). If supplied, the plot is zoomed to the span of every peak whose apex falls within tol of a target, or, for a target with no such peak, whose boundaries (widened by tol) enclose it. The vertical axis is rescaled to the contents of that window, so that small peaks are not flattened against the axis. Defaults to NULL.
tol	Numeric. Half-width of the window around each target, in minutes, regardless of the time unit of chrom_list. Either a single value applied to every target, or a vector giving a separate tolerance for each target. A tol column in targets takes precedence. Defaults to 0.1. Ignored if targets is NULL.
pad	Fraction of the selected peak span to add on either side when zooming. Defaults to 0.5, which keeps neighboring peaks in view. An explicit xlim or ylim passed through ... takes precedence over the zoom; if xlim is given, the vertical axis is rescaled to that window instead.
points	Logical. If TRUE, display peak apex locations as points. Defaults to FALSE.
ticks	Logical. If TRUE, mark peak boundaries with tick marks. Defaults to FALSE.
alpha	Transparency of fitted peak shapes. Defaults to 0.5.
color	Color used to fill fitted peak shapes. If NULL, the color depends on the fitted model type.
cex.points	Size of points. Defaults to 0.5.
numbers	Labels drawn above each peak. TRUE labels peaks with their index in the peak list, and "rt" labels them with their retention time, which can be passed back to targets. Defaults to FALSE.
cex.font	Font size if peaks are numbered. Defaults to 0.5.
digits	Number of decimal places for retention time labels when numbers = "rt". Defaults to 2.
y.offset	Vertical offset of peak numbers, as a fraction of the height of the plot region. Defaults to 0.03.
plot_purity	Whether and how to draw peak purity traces below the chromatogram. "none" (default) draws nothing. "ratio" plots the purity ratio as lollipops on a log scale from 0.01 to 100, clamping values beyond it: blue for pure (ratio < 1), red for impure, with a dotted reference line at ratio = 1. "curves" draws spectral dissimilarity (1 - r ²) against the noise threshold on a fixed [0, 1] scale. The

	threshold is a gray dotted line, and the dissimilarity line is blue where it falls below the threshold (pure) and red where it exceeds it (impure). "curves_local" is the same, but rescales each peak to its own maximum. TRUE maps to "curves" and FALSE to "none".
purity_args	A named list of additional arguments controlling the purity calculation, e.g. list(background_correction = "none"). See get_purity . Arguments that do not apply to the per-peak purity trace drawn here (metric, min_run and try) are ignored, so the same list can be passed to get_peaks and to this function.
res	Time resolution for peak fitting. If missing, inferred from chrom_list.

Details

The appearance of fitted peaks depends on the "fit" attribute of x, which may be "gaussian", "egh", "bemg", or "raw".

Peak lists must contain columns rt, height, start, end, and sd, with additional columns depending on the fit type.

Width parameters are rescaled using the time units of x.

A peak whose shape cannot be drawn is skipped with a warning.

Value

No return value, called for side effects.

Side effects

Plots the trace of chromatogram idx at wavelength lambda, with the fitted peak shapes from x filled in beneath it.

Author(s)

Ethan Bass

See Also

[get_peaks](#)

Other visualization functions: [boxplot.peak_table\(\)](#), [mirror_plot\(\)](#), [plot.chrom_list\(\)](#), [plot.matched_peak_list\(\)](#), [plot.peak_table\(\)](#), [plot_all_spectra\(\)](#), [plot_chrom_contour\(\)](#), [plot_chroms\(\)](#), [plot_chroms_heatmap\(\)](#), [plot_spectrum\(\)](#), [scan_chrom\(\)](#)

Examples

```
data(Sa_warp)
pks <- get_peaks(chrom_list = Sa_warp[1], lambdas = 210)
plot(pks, points = TRUE, ticks = TRUE)
plot(pks, targets = 11.12, ticks = TRUE)
op <- par(mfrow = c(1, 2))
for (t in c(12.19, 14.41)) plot(pks, targets = t)
par(op)
```

plot.peak_table	<i>Plot spectrum from peak table</i>
-----------------	--------------------------------------

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 = NULL,
  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"),
  ...
)
```

Arguments

x	The peak table (output from get_peaktable).
loc	A vector specifying the peak(s) or retention time(s) to plot. If missing when what = "peak", you are prompted for a peak name. When a value is returned (see below), only the first element of loc is used, with a warning.
chrom_list	A list of chromatograms in matrix format (timepoints × wavelengths). If no argument is provided here, the function will try to find the chrom_list object using the pointer in the peak_table.
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. "click" requires engine = "base".
idx	Numerical index of chromatogram you wish to plot; "max" (default) to plot the chromatogram with the largest value for the peak in the peak table; or "all" to plot spectra for all chromatograms.

lambda	The wavelength at which to plot the trace. "max" (default) selects the wavelength at which the spectrum at loc is highest.
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	Character vector of column names in the sample metadata, used as the right-hand side of the boxplot 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, invisibly returns the spectrum as a data.frame. Defaults to FALSE.
verbose	Logical. If TRUE (default), prints the index and name of each chromatogram plotted, with its retention time, wavelength and absorbance.
engine	Plotting backend to use. One of "base", "ggplot", or "plotly". "ggplot2" is also accepted here, for backward compatibility, but "ggplot" is the spelling the other plotting functions take.
...	Additional arguments to boxplot .

Details

Use it to confirm the identity of a peak, or to check that a particular column in the peak table represents a single compound. It also draws simple box plots of the distribution of a peak against variables defined in sample metadata.

When `plot_trace` is TRUE, plots the trace of chromatogram `idx` at wavelength `lambda`, with a dotted red line at the retention time given by `loc`. When `plot_spectrum` is TRUE, plots the spectrum of the same chromatogram at that retention time. When `box_plot` is TRUE, draws a [boxplot](#) of the peak with groups defined by `vars`.

Value

- If `export_spectrum = FALSE` (default), a `plotly` or `ggplot` object, or nothing if `engine == "base"`.
- If `export_spectrum = TRUE`, the spectrum as a `data.frame` with wavelengths as rows and one column per sample giving absorbance at each wavelength (normalized if `scale_spectrum = TRUE`).

Side effects

If `engine == "base"`, plots are rendered to the active graphics device.

Author(s)

Ethan Bass

See Also

Other visualization functions: `boxplot.peak_table()`, `mirror_plot()`, `plot.chrom_list()`, `plot.matched_peak_list()`, `plot.peak_list()`, `plot_all_spectra()`, `plot_chrom_contour()`, `plot_chroms()`, `plot_chroms_heatmap()`, `plot_spectrum()`, `scan_chrom()`

plot.ptw_list	<i>Plot PTW alignments</i>
---------------	----------------------------

Description

Plots `ptw` alignments.

Usage

```
## S3 method for class 'ptw_list'
plot(
  x,
  what = c("traces", "heatmap"),
  engine = c("base", "ggplot", "plotly"),
  lambdas,
  show_legend = TRUE,
  ...
)
```

Arguments

<code>x</code>	A <code>ptw_list</code> object created by <code>correct_rt</code> .
<code>what</code>	What type of plot to return. Either "traces" (default) or "heatmap".
<code>engine</code>	What plotting engine to use. One of "base" (default), "ggplot", or "plotly".
<code>lambdas</code>	Wavelengths to plot, from among those used for warping. Defaults to all of them. The heatmap shows only the first.
<code>show_legend</code>	Logical. Whether to include sample legend.
<code>...</code>	Ignored.

Details

Plots raw and warped chromatograms, one above the other, either as individual traces or as a heatmap, according to the value of `what`. The plot can be produced using either base R graphics, `ggplot2`, or `plotly`, according to the value of the `engine` argument.

Value

A `plotly` or `ggplot` object when `engine = "plotly"` or `engine = "ggplot"`, respectively. No value is returned when `engine = "base"`.

Side effects

If engine == "base", plots are rendered to the active graphics device.

Author(s)

Ethan Bass

Examples

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

plot_all_spectra	<i>Plot all spectra for chosen peak</i>
------------------	---

Description

Plot the spectra of a peak from several chromatograms. Wrapper for [plot_spectrum](#).

Usage

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

Arguments

loc	The peak, retention time, or scan index from which to extract spectral data.
peak_table	A peak_table object created by get_peaktable .
chrom_list	A list of chromatograms in matrix format (timepoints × wavelengths). If no argument is provided here, the function will try to find the chrom_list object using the pointer in the peak_table.
idx	Vector of chromatograms to plot, or "all" (the default).

engine	Plotting backend to use. One of "base", "ggplot", or "plotly". "ggplot2" is also accepted here, for backward compatibility, but "ggplot" is the spelling the other plotting functions take.
plot_spectrum	Logical. If TRUE, plots the spectrum of the chosen peak. Defaults to TRUE.
export_spectrum	Logical. If TRUE (the default), invisibly returns the spectra as a data.frame.
scale_spectrum	Logical. If TRUE (the default), scales each spectrum to the range [0, 1].
overlapping	Logical. If TRUE (the default), plots the spectra on a single plot. Used only when engine = "base".
verbose	Logical. If TRUE, prints the index and name of each chromatogram plotted, with its retention time, wavelength and absorbance. Defaults to FALSE.
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.
peak	The peak argument is deprecated. Please use loc instead.
...	Additional arguments to plot_spectrum.

Value

- If export_spectrum = TRUE (the default), invisibly returns a data.frame with wavelengths as rows and one column per sample encoding absorbance at each wavelength (normalized if scale_spectrum = TRUE).
- If export_spectrum = FALSE, returns a plotly or ggplot object, or nothing if engine == "base".

Side effects

If engine == "base", the spectra are rendered to the active graphics device.

Author(s)

Ethan Bass

See Also

Other visualization functions: `boxplot.peak_table()`, `mirror_plot()`, `plot.chrom_list()`, `plot.matched_peak_list()`, `plot.peak_list()`, `plot.peak_table()`, `plot_chrom_contour()`, `plot_chroms()`, `plot_chroms_heatmap()`, `plot_spectrum()`, `scan_chrom()`

Examples

```
data(Sa_warp)
pks <- get_peaks(Sa_warp, lambda = "220")
pk_tab <- get_peaktable(pks)
plot_all_spectra(loc = "V11", peak_table = pk_tab, overlapping = TRUE)
```

plot_baseline	<i>Plot an estimated baseline</i>
---------------	-----------------------------------

Description

Plots the baseline retained by [preprocess](#) or [correct_baseline](#) over the trace it was estimated from, annotated with the estimator, smoothing parameter and sampling interval that produced it.

Usage

```
plot_baseline(
  x,
  lambda = NULL,
  idx = 1,
  zoom = TRUE,
  col = "red",
  lty = 1,
  lwd = 2,
  add = FALSE,
  annotate = TRUE,
  ...
)
```

Arguments

<code>x</code>	A matrix or vector with an attached "estimated_baseline", or a list of such matrices, as returned by preprocess or correct_baseline with <code>attach_baseline = TRUE</code> .
<code>lambda</code>	Wavelength to plot, as a string or number. Optional for a single-wavelength chromatogram.
<code>idx</code>	Index or name of the chromatogram to plot, if <code>x</code> is a list. Defaults to 1.
<code>zoom</code>	Logical. Whether to scale the y axis to the baseline, letting the peaks run off the top of the panel. At full scale the baseline is usually pressed against the axis, which hides the estimate bending up into a peak. Defaults to TRUE; set FALSE to see the whole trace.
<code>col, lty, lwd</code>	Graphical parameters for the baseline itself. The trace beneath it is always drawn in gray.
<code>add</code>	Logical. Whether to draw the baseline over an existing plot rather than starting a new one, for comparing two estimates on the same trace. The title and annotation are drawn only by the first call, and <code>...</code> is ignored. Defaults to FALSE.
<code>annotate</code>	Logical. Whether to annotate the plot with the estimator, its smoothing parameter and the sampling interval. Set FALSE when overlaying two estimates with <code>add</code> , since a single annotation cannot describe both; label them with legend instead. Defaults to TRUE.
<code>...</code>	Additional arguments to plot , which override the defaults chosen here — <code>main = ""</code> suppresses the title, and an explicit <code>ylim</code> overrides <code>zoom</code> .

Details

The plot uses base graphics. For other plots, extract the baseline with [get_baseline](#).

Value

The annotation label, invisibly, so it can be reused in a [legend](#). The plot is rendered to the active graphics device.

Author(s)

Ethan Bass

See Also

[get_baseline](#), [preprocess](#), [correct_baseline](#)

Examples

```
data(Sa)
x <- preprocess(Sa[1], dim1 = seq(10, 18, by = .01),
                dim2 = seq(200, 318, by = 2), attach_baseline = TRUE,
                show_progress = FALSE, cl = 1)
plot_baseline(x, lambda = 210)
```

plot_chrom_contour	<i>Plot a chromatogram as a contour plot</i>
--------------------	--

Description

Plots a single multi-wavelength chromatogram as an image, with retention time on the x axis, wavelength on the y axis, and color encoding absorbance.

Usage

```
plot_chrom_contour(
  x,
  idx = 1,
  xlim = NULL,
  ylim = NULL,
  scale = c("asinh", "linear"),
  zlim = NULL,
  show_legend = TRUE,
  title = NULL,
  xlab = "",
  ylab = "Wavelength",
  ...
)
```

Arguments

<code>x</code>	A list of chromatograms in matrix format (timepoints $\times$ wavelengths) or a <code>peak_table</code> containing a pointer to a list of chromatograms accessible in the current environment.
<code>idx</code>	Name or numerical index of the chromatogram to plot. Defaults to the first.
<code>xlim</code>	Range of retention times to plot.
<code>ylim</code>	Range of wavelengths to plot.
<code>scale</code>	Color scale: "asinh" (the default) or "linear".
<code>zlim</code>	Range of absorbance values spanned by the color scale. Values outside it are drawn in the end colors. Defaults to the range of the plotted region.
<code>show_legend</code>	Logical. Whether to display a color bar. Defaults to TRUE.
<code>title</code>	Title for plot. Defaults to the name of the chromatogram; "" omits it.
<code>xlab</code>	X label.
<code>ylab</code>	Y label. Defaults to "Wavelength".
<code>...</code>	Additional arguments to <code>graphics::image()</code> , e.g. <code>useRaster = TRUE</code> to speed up rendering of large chromatograms on a regular grid.

Details

With `scale = "asinh"` (the default), color follows $\text{asinh}(x / s)$, where s is 1/100 of the largest absolute value in the plotted region, or in `zlim` when it is given. The scale is linear near zero and logarithmic above s , so peaks up to about 100 times smaller than the tallest one stay visible. The color bar is labeled in absorbance units.

By default each plot is scaled to its own data, so colors cannot be compared between plots. Pass the same `zlim` to every plot to put them on one scale (see examples).

Value

No return value.

Side effects

The plot is rendered to the active graphics device.

Author(s)

Ethan Bass

See Also

Other visualization functions: `boxplot.peak_table()`, `mirror_plot()`, `plot.chrom_list()`, `plot.matched_peak_list()`, `plot.peak_list()`, `plot.peak_table()`, `plot_all_spectra()`, `plot_chroms()`, `plot_chroms_heatmap()`, `plot_spectrum()`, `scan_chrom()`

Examples

```
data(Sa_pr)
plot_chrom_contour(Sa_pr, idx = 1)

# one color scale for every sample
zlim <- range(unlist(Sa_pr))
plot_chrom_contour(Sa_pr, idx = 1, zlim = zlim)
plot_chrom_contour(Sa_pr, idx = 2, zlim = zlim)

# the same, zoomed in
zoomed <- crop_chroms(Sa_pr, rt_range = c(12, 13))
plot_chrom_contour(zoomed, idx = 1, zlim = range(unlist(zoomed)))
```

plot_chroms

Plot traces from list of chromatograms

Description

Visualizes absorbance traces from a list of one- or multi-wavelength chromatograms using base R, ggplot2, or plotly graphics. For multi-wavelength chromatograms, one or more wavelengths can be selected using lambdas. Traces are thinned to time_resolution before plotting. Each chromatogram specified by idx is plotted as a separate trace in its own color.

Usage

```
plot_chroms(
  x,
  lambdas = NULL,
  idx,
  time_resolution = 0.01,
  time_unit = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = "",
  ylab = "Absorbance",
  engine = c("base", "ggplot", "plotly"),
  linewidth = 1,
  show_legend = FALSE,
  legend_position = "topright",
  title = "",
  what = NULL,
  ...
)
```

Arguments

x	A list of chromatograms in matrix format (timepoints × wavelengths) or a peak_table containing a pointer to a list of chromatograms accessible in the current environment.
---	--

lambdas	A character or numeric vector specifying the wavelengths to plot. Required for multi-wavelength chromatograms and optional for one-dimensional ones, where a supplied wavelength must match the column name.
idx	A vector representing the names or numerical indices of the chromatograms to plot. Defaults to all chromatograms.
time_resolution	Desired temporal resolution for plotting (in minutes). Chromatograms are thinned to approximately this interval prior to plotting, which speeds up rendering of large datasets. Defaults to 0.01.
time_unit	Time units of the provided chromatograms: either min, s, or ms. If possible, units will be detected automatically from chromatogram metadata. If time_unit attribute is not present and no argument is provided, the time units will default to minutes.
xlim	Range of x axis values. Unless ylim is given, the y axis is then fitted to the traces within this range.
ylim	Range of y axis values.
xlab	X label.
ylab	Y label. Defaults to "Absorbance".
engine	Plotting backend to use. One of "base", "ggplot", or "plotly".
linewidth	Line width. Defaults to 1.
show_legend	Logical. Whether to display a legend. Defaults to FALSE.
legend_position	Position of legend. Defaults to "topright". For engine = "base", a top or bottom keyword spreads the entries over columns and, unless ylim is given, widens the y-range so the legend clears the traces. For engine = "ggplot", any value accepted by the legend.position argument of <code>ggplot2::theme()</code> ; the default leaves ggplot2's own placement.
title	Title for plot.
what	Which element to plot when each chromatogram in x is itself a list, as returned by <code>chromConverter::read_chroms()</code> with several what values. A single name (e.g. "dad"), or a vector of names for a nested element (e.g. c("instrument", "PMP1, Pressure")). A nested element can also be written as one string split at the first colon, as in "instrument:PMP1, Pressure", unless that string is itself an element name. Defaults to "dad" for such lists; ignored when x is a list of matrices.
...	Additional arguments to <code>matplot</code> (engine = "base"), <code>ggplot2::geom_line()</code> (engine = "ggplot"), or the line list of <code>plotly::plot_ly()</code> (engine = "plotly").

Value

A plotly or ggplot object when engine = "plotly" or engine = "ggplot", respectively. No value is returned when engine = "base".

Side effects

If engine == "base", the plot is rendered to the active graphics device.

Author(s)

Ethan Bass

See Also

Other visualization functions: `boxplot.peak_table()`, `mirror_plot()`, `plot.chrom_list()`, `plot.matched_peak_list()`, `plot.peak_list()`, `plot.peak_table()`, `plot_all_spectra()`, `plot_chrom_contour()`, `plot_chroms_heatmap()`, `plot_spectrum()`, `scan_chrom()`

Examples

```
data(Sa_warp)
plot_chroms(Sa_warp, lambdas = 210)
```

plot_chroms_heatmap	<i>Plot chromatograms as heatmap</i>
---------------------	--------------------------------------

Description

Plots the traces from a list of chromatograms as a heatmap, with one row per chromatogram and color encoding absorbance at a single wavelength. Adapted from [VPdtw: `plot.VPdtw`](#).

Usage

```
plot_chroms_heatmap(
  x,
  idx = NULL,
  lambdas = NULL,
  engine = c("base", "ggplot", "plotly"),
  show_legend = TRUE,
  xlim = NULL,
  zlim = NULL,
  legend_position = "topright",
  title = "",
  show_ylabs = FALSE
)
```

Arguments

<code>x</code>	A list of chromatograms in matrix format (timepoints $\times$ wavelengths) or a <code>peak_table</code> containing a pointer to a list of chromatograms accessible in the current environment.
<code>idx</code>	A vector of names or numerical indices of the chromatograms to plot. Defaults to all chromatograms.
<code>lambdas</code>	A single wavelength to plot. Required for multi-wavelength chromatograms and optional for one-dimensional ones, where a supplied wavelength must match the column name.

engine	Plotting backend to use. One of <code>"base"</code> , <code>"ggplot"</code> , or <code>"plotly"</code> .
show_legend	Logical. Whether to display a color bar. Defaults to TRUE.
xlim	Range of retention times to plot.
zlim	Range of intensities spanned by the color scale. Values outside it are drawn in the end colors. Defaults to the range within xlim. Pass the same zlim to several plots to put them on one scale.
legend_position	Position of legend. Defaults to <code>"topright"</code> . Cannot be repositioned when using the base plotting engine.
title	Title for plot.
show_ylabs	Logical. Whether to label each row with its chromatogram name. Defaults to FALSE.

Value

A plotly or ggplot object when engine = `"plotly"` or engine = `"ggplot"`, respectively. No value is returned when engine = `"base"`.

Side effects

If engine == `"base"`, the plot is rendered to the active graphics device.

Author(s)

Ethan Bass

See Also

Other visualization functions: `boxplot.peak_table()`, `mirror_plot()`, `plot.chrom_list()`, `plot.matched_peak_list()`, `plot.peak_list()`, `plot.peak_table()`, `plot_all_spectra()`, `plot_chrom_contour()`, `plot_chroms()`, `plot_spectrum()`, `scan_chrom()`

Examples

```
data(Sa_warp)
plot_chroms_heatmap(Sa_warp, lambdas = 210)

# zoomed in, on the color scale of the whole run
zlim <- range(sapply(Sa_warp, function(m) m[, "210"]), na.rm = TRUE)
plot_chroms_heatmap(Sa_warp, lambdas = 210, xlim = c(13, 15), zlim = zlim)
```

plot_spectrum	<i>Plot spectrum from peak table</i>
---------------	--------------------------------------

Description

Visualizes chromatographic traces and/or UV spectra for a selected peak, retention time, or scan index from a `peak_table` or list of chromatograms. Spectra can also be exported as a `data.frame` when `export_spectrum = TRUE`.

Usage

```
plot_spectrum(
  loc = NULL,
  peak_table,
  chrom_list = NULL,
  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"),
  ...
)
```

Arguments

<code>loc</code>	The peak, retention time, or scan index from which to extract spectral data.
<code>peak_table</code>	A <code>peak_table</code> object created by get_peaktable .
<code>chrom_list</code>	A list of chromatograms in matrix format (timepoints × wavelengths). If no argument is provided here, the function will try to find the <code>chrom_list</code> object using the pointer in the <code>peak_table</code> .
<code>idx</code>	Index or name of the chromatogram to plot, or "max" (the default) to select the chromatogram with the largest value for <code>loc</code> in <code>peak_table</code> . "max" is available only when <code>what = "peak"</code> .
<code>lambda</code>	The wavelength at which to plot the chromatographic trace, or "max" (the default) for the wavelength of maximum absorbance in the extracted spectrum.
<code>plot_spectrum</code>	Logical. If TRUE, plots the spectrum of the chosen peak. Defaults to TRUE.
<code>plot_trace</code>	Logical. If TRUE, plots the trace of the chosen peak at <code>lambda</code> . Defaults to TRUE.
<code>spectrum_labels</code>	Logical. If TRUE, plots labels on maxima in spectral plot. Defaults to TRUE.
<code>scale_spectrum</code>	Logical. If TRUE, scales spectrum to unit height. Defaults to FALSE.

export_spectrum	Logical. If TRUE, invisibly returns the spectrum as a data.frame. Defaults to FALSE.
verbose	Logical. If TRUE (default), prints the index and name of each chromatogram plotted, with its retention time, wavelength and absorbance.
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. "click" requires engine = "base".
engine	Plotting backend to use. One of "base", "ggplot", or "plotly". "ggplot2" is also accepted here, for backward compatibility, but "ggplot" is the spelling the other plotting functions take.
...	Additional arguments.

Details

Use it to confirm the identity of a peak, or to assess whether a peak table column likely represents a single compound. Retention times may also be selected interactively by clicking on a chromatographic trace when what = "click".

When plot_trace is TRUE, the chromatographic trace for the specified chromatogram (idx) is plotted at wavelength lambda, with a dotted red line indicating the selected retention time (loc). The trace corresponds to a single column of the chromatographic matrix.

When plot_spectrum is TRUE, the UV spectrum at the specified retention time is plotted. The spectrum corresponds to a single row of the chromatographic matrix.

Value

- If export_spectrum = FALSE (default), returns a plotly or ggplot object according to the specified engine. No value is returned when engine = "base".
- If export_spectrum is TRUE, invisibly returns the spectrum as a data.frame with wavelengths as rows and a single column encoding the absorbance (normalized, if scale_spectrum = TRUE).

Side effects

If engine == "base", plots are rendered to the active graphics device.

Author(s)

Ethan Bass

See Also

Other visualization functions: `boxplot.peak_table()`, `mirror_plot()`, `plot.chrom_list()`, `plot.matched_peak_list()`, `plot.peak_list()`, `plot.peak_table()`, `plot_all_spectra()`, `plot_chrom_contour()`, `plot_chroms()`, `plot_chroms_heatmap()`, `scan_chrom()`

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 = "V8", peak_table = pk_tab, what = "peak")
par(oldpar)
```

plot_spectrum_inset	<i>Plot a chromatogram or chromatograms with an inset UV spectrum</i>
---------------------	---

Description

Plots a chromatogram (either a single trace or multiple traces) with an inset UV spectrum for a specified peak. Dotted lines connect the peak apex to the two nearest corners of the inset box.

Usage

```
plot_spectrum_inset(
  loc,
  peak_table,
  chrom_list = NULL,
  idx = "max",
  lambda = "max",
  position = "topright",
  inset_width = 0.35,
  inset_height = 0.35,
  ylim = NULL,
  xlim = NULL,
  engine = "ggplot",
  scale_spectrum = TRUE,
  verbose = getOption("verbose"),
  inset_text_size = 8,
  ...
)
```

Arguments

loc	Peak name or vector of names in the peak table (e.g. "V10").
peak_table	A peak_table object created by get_peaktable .
chrom_list	A list of chromatograms in matrix format (timepoints × wavelengths). If no argument is provided here, the function will try to find the chrom_list object using the pointer in the peak_table.
idx	Index of chromatogram(s) to plot. If "max" (default) or a scalar, plots a single trace, and the inset shows its spectrum. If a vector, plots one trace per chromatogram, and the inset overlays their spectra.

lambda	Wavelength to use for the chromatographic trace and for locating the peak apex. Either a numeric value or "max" (default) to use the wavelength of maximum absorbance.
position	Position of the inset. One of "topright" (default), "topleft", "top", "left", or "right". Alternatively, a numeric vector of length 2 giving the proportional coordinates of the bottom-left corner, e.g., c(0.6, 0.6), used together with inset_width and inset_height. When loc names several peaks, a list with one position per peak.
inset_width	Width of the inset as a proportion of the plot width. Defaults to 0.35.
inset_height	Height of the inset as a proportion of the plot height. Defaults to 0.35.
ylim	Y axis limits for the main plot. Passed to plot_chroms() .
xlim	X axis limits for the main plot. Passed to plot_chroms() .
engine	Plotting engine. Currently only "ggplot" is supported.
scale_spectrum	Logical. If TRUE (the default), scales the inset spectrum to unit height.
verbose	Logical. If TRUE, prints the index and name of each chromatogram plotted, with its retention time, wavelength and absorbance. Defaults to <code>getOption("verbose")</code> .
inset_text_size	Base size for text elements in inset. Defaults to 8.
...	Additional arguments passed to plot_chroms() .

Value

A ggplot object.

See Also

[plot_spectrum\(\)](#), [plot_chroms\(\)](#)

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  data(pk_tab)
  data(Sa_warp)
  # single trace with inset spectrum
  plot_spectrum_inset("V13", peak_table = pk_tab)

  # multiple traces, inset in the center
  plot_spectrum_inset("V13", peak_table = pk_tab, idx=c(1:4),
    position = "top")

  # manual inset placement
  plot_spectrum_inset("V13", peak_table = pk_tab,
    position = c(0.6, 0.3),
    inset_width = 0.35, inset_height = 0.25)

  # multiple peaks with a list of positions
  plot_spectrum_inset(c("V13", "V25"), peak_table = pk_tab,
    position = list(c(0.3, 0.5), c(0.65, 0.3)),
```

```
inset_width = 0.3, inset_height = 0.3)
}
```

preprocess

Preprocess time/wavelength data

Description

Standard preprocessing of time $\times$ wavelength response matrices (e.g. HPLC-DAD/UV data), consisting of: (i) optional subtraction of a reference band of wavelengths, (ii) optional interpolation in either dimension for dimensionality reduction, (iii) optional smoothing along the spectral axis, and (iv) optional baseline correction along the time axis. For densely sampled data (e.g., UV-VIS spectra), the size of the matrix can be reduced by interpolation. By default, the data are smoothed along the spectral axis using cubic smoothing splines ([smooth.spline](#)), then baseline-corrected along the time axis by asymmetric least squares ([asysm](#)). Three other baseline estimators are available through `baseline`, and `baseline = "none"` skips the step; see [baseline_estimators](#).

Usage

```
preprocess(
  X,
  dim1,
  dim2,
  remove.time.baseline = NULL,
  baseline = c("als", "arpls", "aspls", "snip", "none"),
  attach_baseline = FALSE,
  zero_floor = FALSE,
  smooth_spectra = TRUE,
  reference_wavelengths = NULL,
  interpolate_rows = TRUE,
  interpolate_cols = TRUE,
  cl = 2,
  show_progress = NULL,
  outlier_cutoff = 5/60,
  ...,
  maxI = NULL,
  spec.smooth = NULL
)
```

Arguments

<code>X</code>	A numeric matrix (time $\times$ wavelength) or a list of such matrices. Row names must correspond to time points and column names to wavelengths.
<code>dim1</code>	Numeric vector specifying the target time grid, usually built with seq . The data will be interpolated onto these time points, which must be strictly increasing. Chromatograms that do not cover the whole of <code>dim1</code> are dropped with a warning.

	Writing <code>c(from, to, by = x)</code> in place of <code>seq(from, to, by = x)</code> yields a three-element vector rather than a sequence, so it raises an error. If <code>dim1</code> is omitted, the grid runs in steps of 0.01 (in the time units of the data) across the range shared by all chromatograms.
<code>dim2</code>	Numeric vector specifying the target wavelength grid, usually built with seq . The data will be interpolated onto these wavelengths, which must be strictly increasing and lie within the range of the original wavelengths; otherwise an error is raised. Ignored when <code>interpolate_cols = FALSE</code> . If omitted, the existing wavelengths are kept.
<code>remove.time.baseline</code>	The <code>remove.time.baseline</code> argument is deprecated. Please use <code>baseline = "none"</code> instead.
<code>baseline</code>	Baseline estimator to use along the time axis. Either asymmetric least squares (" <code>als</code> ", the default, via asym), asymmetrically reweighted penalized least squares (" <code>arpls</code> "), adaptive smoothness parameter penalized least squares (" <code>aspls</code> "), statistics-sensitive non-linear iterative peak-clipping (" <code>snip</code> "), or " <code>none</code> " to skip baseline correction along the time axis altogether. See baseline_estimators for a description of each.
<code>attach_baseline</code>	Logical. Whether to retain the estimated baseline alongside the corrected data as an " <code>estimated_baseline</code> " attribute. The parameter that produced it (<code>stiffness</code> , or <code>m</code> for " <code>snip</code> ") is always recorded, as a " <code>baseline_param</code> " attribute, whether or not the baseline itself is kept. The baseline is subtracted either way; this argument only controls whether the estimate is kept for inspection. Defaults to <code>FALSE</code> : the baseline is estimated separately for each wavelength, so it is the same size as the data and retaining it roughly doubles memory use. Adding the attribute back to the returned data recovers the uncorrected matrix only when <code>zero_floor = FALSE</code> and <code>maxI</code> and <code>reference_wavelengths</code> are unset: clipping alters the data without altering the stored estimate, <code>maxI</code> rescales both by a factor derived from the clipped data, and the subtracted reference band is not stored.
<code>zero_floor</code>	Logical. Whether to clip negative values in the corrected data to zero. Defaults to <code>FALSE</code> . Clipping discards the half of the noise that falls below zero, which hides the negative dips that show where a baseline was over-subtracted, so it is turned off by default.
<code>smooth_spectra</code>	Logical, indicating whether smoothing should be done along the spectral axis, according to smooth.spline . Skipped for data with fewer than four wavelengths. Defaults to <code>TRUE</code> .
<code>reference_wavelengths</code>	Numeric vector of length 2 giving a band of wavelengths, <code>c(from, to)</code> , whose mean is subtracted from every wavelength at each time point, like the reference wavelength of a diode-array detector. This removes offsets shared by all wavelengths, such as lamp drift or refractive-index effects, before the baseline is estimated. Choose a band where nothing in the samples absorbs. Either bound may be <code>NA</code> to leave that side open. Defaults to <code>NULL</code> (no subtraction).
<code>interpolate_rows</code>	Logical. Whether to interpolate along the time axis (<code>dim1</code>). Defaults to <code>TRUE</code> .

<code>interpolate_cols</code>	Logical. Whether to interpolate along the spectral axis (<code>dim2</code>). Defaults to <code>TRUE</code> .
<code>cl</code>	Either an integer specifying the number of cores to use for parallel processing or a cluster object created by <code>makeCluster</code> . Defaults to 2. On Windows integer values will be ignored.
<code>show_progress</code>	Logical. Whether to show progress bar. Defaults to <code>TRUE</code> if <code>pbapply</code> is installed.
<code>outlier_cutoff</code>	Threshold (in minutes) for excluding chromatograms that end earlier than expected. Samples ending more than this value before the median end time are removed with a warning. The value is always in minutes, whatever time unit the chromatograms are stored in. Defaults to 5/60, i.e. 5 seconds. Only applies when <code>dim1</code> is not specified.
<code>...</code>	Additional arguments to the selected baseline estimator: e.g. <code>stiffness</code> and <code>p</code> for <code>"als"</code> (<code>asysm</code>), <code>stiffness</code> for <code>"arpls"</code> , <code>stiffness</code> and <code>k</code> for <code>"aspls"</code> , or <code>m</code> (required) for <code>"snip"</code> . Because <code>maxI</code> follows <code>...</code> in the signature, it must be named in full: an abbreviation such as <code>max</code> is passed to the estimator, which rejects it.
<code>maxI</code>	If supplied, each chromatogram is rescaled so that its maximum intensity equals <code>maxI</code> .
<code>spec.smooth</code>	The <code>spec.smooth</code> argument is deprecated. Please use <code>smooth_spectra</code> instead.

Details

The operations are applied in the order given above: reference subtraction first, on the original wavelengths, then interpolation, then spectral smoothing, then baseline correction. Because the baseline is estimated after the time axis is interpolated, `stiffness` is tied to whichever `dim1` grid was chosen: the penalty applies on the index axis, so the same value gives a different baseline on a differently sampled grid. Choose `dim1` first, then tune `stiffness` against that grid.

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: <https://github.com/rwehrens/alsace/blob/master/R/preprocess.R>.

Author(s)

Ethan Bass

References

- Baek, S.-J., Park, A., Ahn, Y.-J., and Choo, J. 2015. Baseline correction using asymmetrically reweighted penalized least squares smoothing. *Analyst* 140:250-257. doi:10.1039/C4AN01061B.

- Zhang, F., Tang, X., Tong, A., Wang, B., Wang, J., Lv, Y., Tang, C., and Wang, J. 2020. Baseline correction for infrared spectra using adaptive smoothness parameter penalized least squares method. *Spectroscopy Letters* 53:222-233. doi:10.1080/00387010.2020.1730908.
- Ryan, C.G., Clayton, E., Griffin, W.L., Sie, S.H., and Cousens, D.R. 1988. SNIP, a statistics-sensitive background treatment for the quantitative analysis of PIXE spectra in geoscience applications. *Nuclear Instruments and Methods in Physics Research B* 34:396-402. doi:10.1016/0168583X(88)900638.
- Erb, D. 2025. *pybaselines: A Python library of algorithms for the baseline correction of experimental data*. doi:10.5281/zenodo.5608581.

See Also

[baseline_estimators](#), [correct_baseline](#), [crop_chroms](#), [get_baseline](#)

Other preprocessing functions: [correct_baseline\(\)](#), [crop_chroms\(\)](#), [subtract_blanks\(\)](#)

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)
```

```
print.matched_peak_list
```

Print matched peak list

Description

Prints a summary of a matched_peak_list: the wavelength and reference sample, the matching settings, and how many peaks each sample matched.

Usage

```
## S3 method for class 'matched_peak_list'
print(x, ...)
```

Arguments

x	A matched_peak_list object.
...	Additional arguments (ignored).

Value

Invisibly returns x.

Author(s)

Ethan Bass

```
print.matched_peak_list_set
```

Print a set of matched peak lists

Description

Prints a summary of a matched_peak_list_set: its wavelengths, the reference sample, and the reference peaks and targets at each wavelength.

Usage

```
## S3 method for class 'matched_peak_list_set'  
print(x, ...)
```

Arguments

x	A matched_peak_list_set object.
...	Additional arguments (ignored).

Value

Invisibly returns x.

Author(s)

Ethan Bass

```
print.peak_list
```

Print peak list

Description

Prints a summary of a peak_list object, including the number of samples, wavelengths, fit method, and total number of peaks.

Usage

```
## S3 method for class 'peak_list'  
print(x, ...)
```

Arguments

x	A peak_list object.
...	Additional arguments (ignored).

Value

Invisibly returns `x`.

Author(s)

Ethan Bass

reshape_chroms	<i>Reshape chromatograms</i>
----------------	------------------------------

Description

Converts a list of chromatographic matrices into a single long-format data frame with one row per sample $\times$ retention time $\times$ wavelength combination.

Usage

```
reshape_chroms(  
  x,  
  idx,  
  time_resolution = NULL,  
  time_unit = NULL,  
  sample_var = "sample",  
  lambdas = NULL,  
  rts = NULL,  
  transfer_metadata = FALSE  
)
```

Arguments

<code>x</code>	A list of chromatographic matrices in wide format.
<code>idx</code>	Indices of chromatograms to convert. Defaults to all of them.
<code>time_resolution</code>	Desired temporal resolution, in minutes, regardless of the time unit of <code>x</code> , matching <code>plot_chroms</code> . Subsamples the time axis to approximately this interval. Defaults to <code>NULL</code> , which keeps the full resolution.
<code>time_unit</code>	Time units of the provided chromatograms: either <code>"min"</code> , <code>"s"</code> , or <code>"ms"</code> . Used only to interpret <code>time_resolution</code> . Defaults to <code>NULL</code> , in which case the unit is taken from the <code>time_unit</code> attribute of the chromatograms, falling back to <code>"min"</code> .
<code>sample_var</code>	Name of the new column containing sample identifiers.
<code>lambdas</code>	Wavelength(s) to include. Defaults to all wavelengths.
<code>rts</code>	Retention times to include, each matched to the nearest time point. Defaults to all retention times.
<code>transfer_metadata</code>	Logical. If <code>TRUE</code> , metadata attributes are transferred to the output. Defaults to <code>FALSE</code> .

Value

A data frame in long format with columns `rt`, `lambda`, `absorbance`, and a sample identifier column specified by `sample_var`.

Author(s)

Ethan Bass

See Also

Other utility functions: [combine_peaks\(\)](#), [merge_peaks\(\)](#), [reshape_peaktable\(\)](#)

Examples

```
data(Sa_warp)
reshape_chroms(Sa_warp, lambdas = 210)
```

reshape_peaktable	<i>Reshape peaktable</i>
-------------------	--------------------------

Description

Reshapes a peak table to long format, or to a wide data.frame with the selected peaks alongside sample metadata.

Usage

```
reshape_peaktable(
  x,
  peaks,
  metadata,
  fixed_levels = TRUE,
  format = c("long", "wide")
)
```

Arguments

<code>x</code>	A <code>peak_table</code> object.
<code>peaks</code>	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.
<code>metadata</code>	A character vector specifying the metadata fields to include.
<code>fixed_levels</code>	Logical. Whether to fix factor levels of features in the order provided. Defaults to TRUE. Ignored when <code>format = "wide"</code> .
<code>format</code>	Either "long" (the default), with one row per sample and peak, or "wide", with one row per sample and one column per peak.

Value

A data.frame containing the information for the specified peaks. In long format it has columns sample, peak, lambda, rt, the response and any metadata; in wide format, sample, any metadata, then one column per peak.

Author(s)

Ethan Bass

See Also

Other utility functions: [combine_peaks\(\)](#), [merge_peaks\(\)](#), [reshape_chroms\(\)](#)

Examples

```
data(pk_tab)
reshape_peaktable(pk_tab, peaks = c("V8", "V13"))
reshape_peaktable(pk_tab, peaks = c(a = "V8", b = "V13"), format = "wide")
```

Sa

Raw goldenrod root chromatograms

Description

A list of four HPLC-DAD data matrices of *Solidago altissima* roots extracted in 90\ wavelengths are stored in columns. Data were collected on an Agilent 1100 HPLC.

Usage

```
data(Sa)
```

Format

A list of four matrices (1301 timepoints $\times$ 60 wavelengths).

See Also

Other data objects: [Sa_pr](#), [Sa_warp](#), [pk_tab](#)

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 interpolated from the 0.0067-minute interval of [Sa](#) to 0.02 minutes, to save space and processing time.

Usage

```
data(Sa_pr)
```

Format

A list of four pre-processed matrices (434 timepoints $\times$ 60 wavelengths).

See Also

Other data objects: [Sa](#), [Sa_warp](#), [pk_tab](#)

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 timepoints $\times$ 60 wavelengths).

See Also

Other data objects: [Sa](#), [Sa_pr](#), [pk_tab](#)

scan_chrom

*Plot spectra by clicking on the chromatogram***Description**

Plots a chromatographic trace from the specified chromatogram (`idx`), 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.

Usage

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

Arguments

<code>chrom_list</code>	A list of chromatograms in matrix format (timepoints × wavelengths). If no argument is provided here, the function will try to find the <code>chrom_list</code> object using the pointer in the <code>peak_table</code> .
<code>idx</code>	Index or name of the chromatogram to plot. If missing, you are prompted for one.
<code>lambda</code>	The wavelength at which to plot the chromatographic trace. Required; "max" is not accepted.
<code>plot_spectrum</code>	Logical. If TRUE, plots the spectrum of the chosen peak. Defaults to TRUE.
<code>peak_table</code>	A <code>peak_table</code> object created by get_peaktable .
<code>scale_spectrum</code>	Logical. If TRUE, scales spectrum to unit height. Defaults to FALSE.
<code>spectrum_labels</code>	Logical. If TRUE, plots labels on maxima in spectral plot. Defaults to TRUE.
<code>export_spectrum</code>	Logical. If TRUE, invisibly returns the spectrum as a <code>data.frame</code> . Defaults to FALSE.
<code>...</code>	Additional arguments to <code>plot_spectrum</code> .

Details

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

Value

Invisibly returns the spectrum when `export_spectrum = TRUE` and `plot_spectrum = TRUE`; otherwise the numeric index of the scan selected by the user.

Side effects

Plots are rendered to the active graphics device.

Author(s)

Ethan Bass

See Also

Other visualization functions: `boxplot.peak_table()`, `mirror_plot()`, `plot.chrom_list()`, `plot.matched_peak_list()`, `plot.peak_list()`, `plot.peak_table()`, `plot_all_spectra()`, `plot_chrom_contour()`, `plot_chroms()`, `plot_chroms_heatmap()`, `plot_spectrum()`

Examples

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

subtract_blanks	<i>Subtract blank chromatograms</i>
-----------------	-------------------------------------

Description

Subtracts a blank chromatogram from each sample in a `chrom_list`, using either the mean of all blanks, the nearest blank in run order, or the most recent preceding blank.

Usage

```
subtract_blanks(  
  chrom_list,  
  pattern = NULL,  
  blank_id,  
  method = c("mean", "nearest", "preceding"),  
  zero_floor = FALSE  
)
```

Arguments

<code>chrom_list</code>	A <code>chrom_list</code> object.
<code>pattern</code>	A regular expression matched against <code>names(chrom_list)</code> to identify blanks (e.g. <code>"^Blank"</code> for samples named Blank1, Blank2, etc.). Takes precedence over <code>blank_id</code> if both are supplied.
<code>blank_id</code>	Names or indices of the elements in <code>chrom_list</code> that are blanks. Ignored if <code>pattern</code> is supplied.
<code>method</code>	How to select the blank to subtract for each sample: <code>"mean"</code> subtracts the elementwise mean of all blanks from every sample; <code>"nearest"</code> subtracts the blank closest in run order (list position), taking the earlier one when two are equally close; <code>"preceding"</code> subtracts the closest blank occurring before the sample in run order, and raises an error if a sample has none.
<code>zero_floor</code>	Logical. If TRUE, values below zero after subtraction are set to zero, with a warning if this affects more than 5% of a sample's values. Defaults to FALSE: a blank that matches the samples' baseline leaves them scattered around zero, and flooring would discard that signal before baseline correction.

Details

Blanks recorded on a different time grid from a sample (for example, one or two points longer or shorter) are linearly interpolated onto the sample's retention times before subtraction. Where the sample runs past either end of the blank, the blank's first or last value is carried forward, with a warning if this spans more than 5 time points. Blanks and samples must share the same wavelengths.

Value

A `chrom_list` with blank-subtracted samples. Blanks themselves are removed from the returned list, and so is any `"estimated_baseline"` attribute on the samples.

See Also

[preprocess](#), [correct_baseline](#)

Other preprocessing functions: [correct_baseline\(\)](#), [crop_chroms\(\)](#), [preprocess\(\)](#)

Examples

```
data(Sa_warp)
subtract_blanks(Sa_warp, blank_id = 1)
```

targeted_analysis	<i>Targeted analysis</i>
-------------------	--------------------------

Description

Targeted analysis measures a set of known compounds. Two routes lead from a table of target compounds to a peak table with a named column for each target: selecting peaks by retention time in [get_peaks](#), or matching peaks to a reference sample with [match_peaks](#).

Choosing a route

Select peaks by retention time using [get_peaks](#) when the samples are aligned, or can be aligned with [correct_rt](#), and no other peak falls within `tol` of a target. Only the peaks near the targets are fitted, which is quicker, and each target can be measured at its own wavelength.

Match peaks to a reference using [match_peaks](#) when retention times alone cannot identify the targets (e.g., in crowded windows). Peaks are matched by spectrum as well as retention time, so a target can be picked out from its neighbors, and the retention-time window can be wider to allow for drift. Every peak is fitted, and every target is measured at the same wavelength.

Constructing the targets table

Build a `data.frame` with an `rt` column giving the retention time of each target and, optionally:

- `name`: Unique names, used as peak table column names.
- `lambda`: The wavelength at which to measure each target. When matching to a reference, it must equal the `lambda` given to [match_peaks](#).
- `tol`: How far each target's peak may lie from its `rt`, in minutes, regardless of the time unit of the chromatograms. Defaults to `0.1`.
- `spectral_cutoff`: Used only by [match_peaks](#). The minimum spectral correlation for a match to each target's reference peak. Defaults to the `spectral_cutoff` argument of [match_peaks](#).
- `rt_window`: Used only by [match_peaks](#). The maximum difference in retention time, in minutes, between each target's reference peak and its match in another sample. Defaults to the `rt_window` argument of [match_peaks](#).

Selecting peaks by retention time

1. Call `get_peaks(chrom_list, targets = targets)`. In each sample, `get_peaks` fits only the peaks whose apex lies within `tol` of a target.
2. Call [get_peaktable](#) on the result. The peak table has one column per target found in at least one sample; a target found in no sample gets no column.

Peaks are assigned by retention time only, so each target must stay within `tol` of its listed time in every sample, and closer to it than to any neighboring target. If the samples drift more than that, fit warping models with [correct_rt](#) and pass them as a `mod_list`; the targets are then found on the aligned time axis.

When more than one peak falls within `tol` of a target, the `select_by` argument of `get_peaks` decides which is retained. The default, `select_by = "height"`, keeps the tallest, which may not be the target; narrow `tol`, or set `select_by = "proximity"` to keep the peak closest to the target. Peaks are selected before they are fitted, so "height" compares the signal at each apex, including any baseline beneath it.

To keep every candidate in the peak list, set `select_by = "none"`. Where a sample has several peaks for one target, `get_peaktable` enters the one with the largest response in the table, but summarizes the target's `pk_meta` over all of them, so its `rt` can fall between the candidates. To take the metadata from the most intense peak instead, set `summarize_by = "max"` in `get_peaktable`.

Matching peaks to a reference sample

1. Call `get_peaks` without providing targets, so that every peak is fitted.
2. Call `match_peaks(peak_list, targets = targets, lambda = ...)`. Unless reference is supplied, `match_peaks` picks as reference a sample that has a peak for every target, then matches each sample's peaks to the reference peaks by spectral correlation and retention time.
3. Call `get_peaktable(matched, pks = peak_list)`. Unlike the other pathway, the peak table will still include untargeted peaks. Use `[, targets$name]` to keep only the targets.

In the reference sample, each target is assigned to the tallest well-fitted peak within `tol` minutes of the specified `rt`. A peak in another sample can match a reference peak only if it lies within `rt_window` minutes of it and their spectral correlation is above the `spectral_cutoff`.

Author(s)

Ethan Bass

See Also

[get_peaks](#), [match_peaks](#), [get_peaktable](#)

Examples

```
data(Sa_pr)
targets <- data.frame(name = c("A", "B"), rt = c(12.07, 14.31))

# selecting by retention time, on the aligned time axis
mods <- correct_rt(Sa_pr, lambdas = 220, what = "models", reference = 1)
pks_t <- get_peaks(Sa_pr, lambdas = 220, targets = targets, mod_list = mods)
get_peaktable(pks_t)

# matching to a reference sample
pks <- get_peaks(Sa_pr, lambdas = 220)
matched <- match_peaks(pks, targets = targets, lambda = 220)
get_peaktable(matched, pks = pks)[, targets$name]
```

write_peaktable	<i>Export peak table</i>
-----------------	--------------------------

Description

Exports peak table in csv or xlsx format.

Usage

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

Arguments

peak_table	Peak table object from get_peaktable .
path	Path to an existing directory in which to write the files.
filename	File name. Defaults to "peak_table".
format	File format to export. Either "csv" (default) or "xlsx".
what	Which elements of the peak_table to export. One or more of "tab", "pk_meta", "sample_meta", "instrument_meta", "ref_spectra", and "args". Defaults to all six.

Value

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

Side effects

With format = "csv", writes one file per element of what, named <filename>-<element>.csv.
With format = "xlsx", writes a single <filename>.xlsx workbook with one sheet per element.
Requires the openxlsx package.

Examples

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

Index

- * **accessors**
 - get_baseline, 34
 - get_lambdas, 35
 - get_time_resolution, 47
 - get_times, 48
- * **data objects**
 - pk_tab, 57
 - Sa, 86
 - Sa_pr, 87
 - Sa_warp, 87
- * **data**
 - pk_tab, 57
 - Sa, 86
 - Sa_pr, 87
 - Sa_warp, 87
- * **preprocessing functions**
 - correct_baseline, 19
 - crop_chroms, 30
 - preprocess, 79
 - subtract_blanks, 89
- * **utility functions**
 - combine_peaks, 17
 - merge_peaks, 52
 - reshape_chroms, 84
 - reshape_peaktable, 85
- * **visualization functions**
 - boxplot.peak_table, 14
 - mirror_plot, 53
 - plot.chrom_list, 58
 - plot.matched_peak_list, 59
 - plot.peak_list, 60
 - plot.peak_table, 63
 - plot_all_spectra, 66
 - plot_chrom_contour, 69
 - plot_chroms, 71
 - plot_chroms_heatmap, 73
 - plot_spectrum, 75
 - scan_chrom, 88
 - [.matched_peak_list, 6
 - [.peak_list, 7, 41
 - [.peak_table, 8, 57
 - annotate_peaks, 9
 - asym, 12, 19, 20, 79–81
 - attach_metadata, 10, 44, 46, 55, 56
 - attach_ref_spectra, 11, 16, 44, 46, 56
 - baseline_estimators, 12, 19, 20, 79, 80, 82
 - bestref, 23, 27, 49
 - boxplot, 14, 64
 - boxplot.peak_table, 14
 - boxplot.peak_table(), 54, 58, 60, 62, 65, 67, 70, 73, 74, 76, 89
 - chromatographR
 - (chromatographR-package), 3
 - chromatographR-package, 3
 - chromConverter::read_chroms(), 58, 72
 - cluster_spectra, 15
 - combine_peaks, 17
 - combine_peaks(), 53, 85, 86
 - correct_baseline, 12–14, 19, 30, 31, 34, 68, 69, 82, 90
 - correct_baseline(), 31, 82, 90
 - correct_peaks, 21, 26, 29, 37, 45, 50
 - correct_rt, 21, 22, 22, 29, 34, 37, 49, 50, 52, 65, 91
 - correct_rt_group, 26, 26
 - crop_chroms, 30, 34, 82
 - crop_chroms(), 20, 82, 90
 - cutreeDynamicTree, 43
 - dilation, 24
 - dim.peak_table(), 56
 - filter_peaks, 31, 34, 37, 44
 - filter_peaktable, 32, 33, 44
 - find_peaks, 38, 40, 41
 - fit_peaks, 37, 38, 41
 - formula, 14

get_baseline, [14](#), [19](#), [20](#), [30](#), [34](#), [69](#), [82](#)
 get_baseline(), [35](#), [48](#)
 get_lambdas, [30](#), [31](#), [35](#)
 get_lambdas(), [35](#), [48](#)
 get_peaks, [12](#), [18](#), [21](#), [23](#), [29](#), [31](#), [32](#), [36](#), [43](#),
 [44](#), [49](#), [52](#), [62](#), [91](#), [92](#)
 get_peaktable, [9–12](#), [16](#), [18](#), [21](#), [32–34](#), [37](#),
 [40](#), [41](#), [44](#), [50](#), [52](#), [53](#), [55](#), [56](#), [60](#), [63](#),
 [66](#), [75](#), [77](#), [88](#), [91–93](#)
 get_peaktable(), [57](#)
 get_purity, [37](#), [39](#), [62](#)
 get_spectra, [38](#), [46](#)
 get_time_resolution, [47](#)
 get_time_resolution(), [35](#), [48](#)
 get_times, [30](#), [31](#), [48](#)
 get_times(), [35](#), [48](#)
 ggplot2::annotate(), [10](#)
 ggplot2::geom_line(), [72](#)
 ggplot2::theme(), [72](#)
 graphics::image(), [70](#)

 head.peak_table(), [56](#)

 legend, [68](#), [69](#)

 makeCluster, [19](#), [25](#), [29](#), [37](#), [81](#)
 match(), [38](#)
 match_peaks, [23](#), [24](#), [28](#), [43](#), [45](#), [49](#), [59](#), [60](#),
 [91](#), [92](#)
 matplot, [54](#), [72](#)
 merge_peaks, [52](#)
 merge_peaks(), [18](#), [85](#), [86](#)
 mirror_plot, [53](#)
 mirror_plot(), [15](#), [58](#), [60](#), [62](#), [65](#), [67](#), [70](#), [73](#),
 [74](#), [76](#), [89](#)

 normalize_data, [11](#), [55](#)

 pbapply, [20](#), [25](#), [28](#), [37](#), [81](#)
 peak_table, [8](#), [44](#), [46](#)
 peak_table (peak_table-class), [56](#)
 peak_table-class, [56](#)
 pk_tab, [57](#), [86](#), [87](#)
 plot, [59](#), [68](#)
 plot.chrom_list, [58](#)
 plot.chrom_list(), [15](#), [54](#), [60](#), [62](#), [65](#), [67](#),
 [70](#), [73](#), [74](#), [76](#), [89](#)
 plot.matched_peak_list, [50](#), [59](#)
 plot.matched_peak_list(), [15](#), [54](#), [58](#), [62](#),
 [65](#), [67](#), [70](#), [73](#), [74](#), [76](#), [89](#)

 plot.matched_peak_list_set
 (plot.matched_peak_list), [59](#)
 plot.peak_list, [60](#)
 plot.peak_list(), [15](#), [54](#), [58](#), [60](#), [65](#), [67](#), [70](#),
 [73](#), [74](#), [76](#), [89](#)
 plot.peak_table, [63](#)
 plot.peak_table(), [15](#), [54](#), [58](#), [60](#), [62](#), [67](#),
 [70](#), [73](#), [74](#), [76](#), [89](#)
 plot.ptw_list, [65](#)
 plot_all_spectra, [66](#)
 plot_all_spectra(), [15](#), [54](#), [58](#), [60](#), [62](#), [65](#),
 [70](#), [73](#), [74](#), [76](#), [89](#)
 plot_baseline, [12](#), [14](#), [68](#)
 plot_chrom_contour, [69](#)
 plot_chrom_contour(), [15](#), [54](#), [58](#), [60](#), [62](#),
 [65](#), [67](#), [73](#), [74](#), [76](#), [89](#)
 plot_chroms, [47](#), [71](#), [84](#)
 plot_chroms(), [9](#), [10](#), [15](#), [54](#), [58](#), [60](#), [62](#), [65](#),
 [67](#), [70](#), [74](#), [76](#), [78](#), [89](#)
 plot_chroms_heatmap, [73](#)
 plot_chroms_heatmap(), [15](#), [54](#), [58](#), [60](#), [62](#),
 [65](#), [67](#), [70](#), [73](#), [76](#), [89](#)
 plot_spectrum, [8](#), [66](#), [75](#)
 plot_spectrum(), [15](#), [54](#), [58](#), [60](#), [62](#), [65](#), [67](#),
 [70](#), [73](#), [74](#), [78](#), [89](#)
 plot_spectrum_inset, [77](#)
 plot_spectrum_inset(), [10](#)
 plotly::plot_ly(), [72](#)
 preprocess, [12–14](#), [20](#), [23](#), [24](#), [27](#), [28](#), [30](#), [31](#),
 [34](#), [40](#), [47](#), [68](#), [69](#), [79](#), [90](#)
 preprocess(), [20](#), [31](#), [90](#)
 print.matched_peak_list, [82](#)
 print.matched_peak_list_set, [83](#)
 print.peak_list, [41](#), [83](#)
 print.peak_table(), [56](#)
 ptw, [22–26](#), [28](#), [29](#), [65](#)
 pvclust, [16](#), [17](#)
 vvpick, [16](#)
 vprect, [16](#)

 reshape_chroms, [47](#), [84](#)
 reshape_chroms(), [18](#), [53](#), [86](#)
 reshape_peaktable, [85](#)
 reshape_peaktable(), [18](#), [53](#), [85](#)
 row.names.peak_table(), [57](#)

 Sa, [57](#), [86](#), [87](#)
 Sa_pr, [57](#), [86](#), [87](#), [87](#)
 Sa_warp, [57](#), [86](#), [87](#), [87](#)

scan_chrom, 88
scan_chrom(), 15, 54, 58, 60, 62, 65, 67, 70,
73, 74, 76
select.traces, 24, 28
seq, 79, 80
smooth.spline, 79, 80
subset.peak_table(), 57
subtract_blanks, 19, 20, 34, 89
subtract_blanks(), 20, 31, 82
summary.peak_table(), 57

tail.peak_table(), 56
targeted_analysis, 37, 41, 45, 49, 52, 61, 91

VPdtw, 22, 26
VPdtw::plot.VPdtw, 73

write_peaktable, 93