Package 'alkahest'

October 12, 2022

Title Pre-Processing XY Data from Experimental Methods

Version 1.0.0

Maintainer Nicolas Frerebeau <nicolas.frerebeau@u-bordeaux-montaigne.fr>

Description A lightweight, dependency-free toolbox for pre-processing XY data from experimental methods (i.e. any signal that can be measured along a continuous variable). This package provides methods for baseline estimation and correction, smoothing, normalization, integration and peaks detection. Baseline correction methods includes polynomial fitting as described in Lieber and Mahadevan-Jansen (2003) <doi:10.1366/000370203322554518>, Rolling Ball algorithm after Kneen and Annegarn (1996) <doi:10.1016/0168-583X(95)00908-6>, SNIP algorithm after Ryan et al. (1988) <doi:10.1016/0168-583X(88)90063-8>, 4S Peak Filling after Liland (2015) <doi:10.1016/j.mex.2015.02.009> and more.

License GPL (>= 3)

URL https://packages.tesselle.org/alkahest/, https://github.com/tesselle/alkahest

BugReports https://github.com/tesselle/alkahest/issues

Depends R (>= 3.3)

Encoding UTF-8

RoxygenNote 7.2.1

Imports grDevices, methods, stats, utils

Suggests knitr, Matrix, rmarkdown, testthat (>= 3.0.0)

Collate 'AllGenerics.R' 'alkahest-package.R' 'baseline.R' 'data.R' 'helpers.R' 'integrate.R' 'peaks.R' 'replace.R' 'resample.R' 'rescale.R' 'signal.R' 'smooth.R' 'windows.R'

LazyData true

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Nicolas Frerebeau [aut, cre] (<https://orcid.org/0000-0001-5759-4944>,
Université Bordeaux Montaigne),
Brice Lebrun [ctb] (<https://orcid.org/0000-0001-7503-8685>)

Repository CRAN

Date/Publication 2022-09-15 10:00:02 UTC

R topics documented:

baseline_linear . . . 3
baseline_peakfilling . . . 4
baseline_polynomial . . . 5
baseline_rollingball . . . 6
baseline_rubberband . . . 8
baseline_snip . . . 9
BEGe . . . 11
integrate_rectangle . . . 11
integrate_trapezoid . . . 12
LaBr . . . 13
peaks_find . . . 14
peaks_fwhm . . . 16
replace_negative . . . 17
replace_threshold . . . 18
resample_bin . . . 19
resample_down . . . 20
resample_interpolate . . . 22
rescale_area . . . 23
rescale_range . . . 24
rescale_total . . . 26
rescale_transform . . . 27
signal_bind . . . 28
signal_correct . . . 30
signal_drift . . . 31
signal_mean . . . 32
signal_shift . . . 33
smooth_loess . . . 34
smooth_rectangular . . . 36
smooth_savitzky . . . 37
smooth_triangular . . . 39
smooth_whittaker . . . 40
subset . . . 42
window_sliding . . . 43
window_tumbling . . . 45
XRD . . . 46

Index **48**

baseline_linear *Linear Baseline Estimation*

Description

Linear Baseline Estimation

Usage

```
baseline_linear(x, y, ...)

## S4 method for signature 'numeric,numeric'
baseline_linear(x, y, points = range(x))

## S4 method for signature 'ANY,missing'
baseline_linear(x, points = range(x))
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Currently not used.
points	A numeric vector specifying the data points to be used in the fitting process (in x unit).

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

signal_correct()

Other baseline estimation methods: baseline_peakfilling(), baseline_polynomial(), baseline_rollingball(), baseline_rubberband(), baseline_snip()

Examples

```
## X-ray diffraction
data("XRD")

## Plot spectrum
plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")
```

```
## Linear baseline
baseline <- baseline_linear(XRD, points = c(25, 34))

plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")
lines(baseline, type = "l", col = "red")

## Correct baseline
XRD$count <- XRD$count - baseline$y

plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")
```

`baseline_peakfilling` *4S Peak Filling*

Description

Baseline estimation by iterative mean suppression.

Usage

```
baseline_peakfilling(x, y, ...)

## S4 method for signature 'numeric,numeric'
baseline_peakfilling(x, y, n, m, by = 10)

## S4 method for signature 'ANY,missing'
baseline_peakfilling(x, n, m, by = 10)
```

Arguments

`x, y`	A `numeric` vector. If `y` is missing, an attempt is made to interpret `x` in a suitable way (see `grDevices::xy.coords()`).
`...`	Currently not used.
`n`	An `integer` value giving the number of iterations.
`m`	An odd `integer` giving the half window size.
`by`	An `integer` specifying the binning ratio (i.e. the number of points to be grouped together; see `window_tumbling()`).

Author(s)

N. Frerebeau

References

Liland, K. H. (2015). 4S Peak Filling - baseline estimation by iterative mean suppression. *MethodsX*, 2, 135-140. doi:10.1016/j.mex.2015.02.009.

See Also

signal_correct()

Other baseline estimation methods: baseline_linear(), baseline_polynomial(), baseline_rollingball(), baseline_rubberband(), baseline_snip()

Examples

```
## gamma-ray spectrometry
data("BEGe")

## Subset from 2.75 to 200 keV
BEGe <- signal_select(BEGe, from = 3, to = 200)

## Plot spectrum
plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")

## 4S Peak Filling
baseline <- baseline_peakfilling(BEGe, n = 5, m = 5, by = 2)

plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")
lines(baseline, type = "l", col = "red")
```

---

baseline_polynomial *Polynomial Baseline Estimation*

Description

Polynomial Baseline Estimation

Usage

```
baseline_polynomial(x, y, ...)

## S4 method for signature 'numeric,numeric'
baseline_polynomial(x, y, d = 3, tolerance = 0.001, stop = 100)

## S4 method for signature 'ANY,missing'
baseline_polynomial(x, d = 3, tolerance = 0.001, stop = 100)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Currently not used.
d	An integer giving the degree of the polynomial. Must be less than the number of unique points.
tolerance	A numeric scalar giving the tolerance of difference between iterations.
stop	An integer giving the stopping rule (i.e. maximum number of iterations).

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

References

Lieber, C. A. and Mahadevan-Jansen, A. (2003). Automated Method for Subtraction of Fluorescence from Biological Raman Spectra. *Applied Spectroscopy*, 57(11): 1363-67. doi:10.1366/000370203322554518.

See Also

signal_correct()

Other baseline estimation methods: baseline_linear(), baseline_peakfilling(), baseline_rollingball(), baseline_rubberband(), baseline_snip()

Examples

```
## X-ray diffraction
data("XRD")

## Subset from 20 to 70 degrees
XRD <- signal_select(XRD, from = 20, to = 70)

## Plot spectrum
plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")

## Polynomial baseline
baseline <- baseline_polynomial(XRD, d = 4, tolerance = 0.02, stop = 1000)

plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")
lines(baseline, type = "l", col = "red")
```

---

baseline_rollingball *Rolling Ball Baseline Estimation*

Description

Rolling Ball Baseline Estimation

Usage

```
baseline_rollingball(x, y, ...)

## S4 method for signature 'numeric,numeric'
baseline_rollingball(x, y, m, s)

## S4 method for signature 'ANY,missing'
baseline_rollingball(x, m, s)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Currently not used.
m	An odd integer giving the window size (i.e. the number of adjacent points to be used; see window_sliding()) for minimization/maximization.
s	An odd integer giving the window size (i.e. the number of adjacent points to be used; see window_sliding()) for smoothing.

Value

Returns a list with two components x and y.

Note

There will be $(m-1)/2$ points both at the beginning and at the end of the data series for which a complete m-width window cannot be obtained. To prevent data loss, progressively wider/narrower windows are used at both ends of the data series.

Author(s)

N. Frerebeau

References

Kneen, M. A. and Annegarn, H. J. (1996). Algorithm for Fitting XRF, SEM and PIXE X-Ray Spectra Backgrounds. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 109/110: 209-213. doi:10.1016/0168583X(95)009086.

See Also

signal_correct()

Other baseline estimation methods: baseline_linear(), baseline_peakfilling(), baseline_polynomial(), baseline_rubberband(), baseline_snip()

Examples

```
## X-ray diffraction
data("XRD")

## Subset from 20 to 70 degrees
XRD <- signal_select(XRD, from = 20, to = 70)

## Plot spectrum
plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")

## Rolling Ball baseline
baseline <- baseline_rollingball(XRD, m = 201, s = 151)

plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")
lines(baseline, type = "l", col = "red")
```

---

baseline_rubberband *Rubberband Baseline Estimation*

Description

Rubberband Baseline Estimation

Usage

```
baseline_rubberband(x, y, ...)

## S4 method for signature 'numeric,numeric'
baseline_rubberband(x, y, noise = 0, spline = TRUE, ...)

## S4 method for signature 'ANY,missing'
baseline_rubberband(x, noise = 0, spline = TRUE, ...)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Extra arguments to be passed to stats::smooth.spline().
noise	A length-one numeric vector giving the noise level. Only used if method is "rubberband".
spline	A logical scalar: should spline interpolation through the support points be used instead of linear interpolation? Only used if method is "rubberband".

Details

A convex envelope of the spectrum is determined and the baseline is estimated as the part of the convex envelope lying below the spectrum. Note that the rubber band does not enter the concave regions (if any) of the spectrum.

Value

Returns a list with two components x and y.

Note

baseline_rubberband() is slightly modified from C. Beleites' hyperSpec::spc.rubberband().

Author(s)

N. Frerebeau

See Also

signal_correct()

Other baseline estimation methods: baseline_linear(), baseline_peakfilling(), baseline_polynomial(), baseline_rollingball(), baseline_snip()

Examples

```
## gamma-ray spectrometry
data("BEGe")

## Subset from 2.75 to 200 keV
BEGe <- signal_select(BEGe, from = 3, to = 200)

## Plot spectrum
plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")

## Rubberband baseline
baseline <- baseline_rubberband(BEGe)

plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")
lines(baseline, type = "l", col = "red")
```

---

baseline_snip *SNIP Baseline Estimation*

Description

Sensitive Nonlinear Iterative Peak clipping algorithm.

Usage

```
baseline_snip(x, y, ...)

## S4 method for signature 'numeric,numeric'
baseline_snip(x, y, LLS = FALSE, decreasing = FALSE, n = 100)

## S4 method for signature 'ANY,missing'
baseline_snip(x, LLS = FALSE, decreasing = FALSE, n = 100)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Currently not used.
LLS	A logical scalar: should the LLS operator be applied on x before employing SNIP algorithm? Only used if method is "SNIP".
decreasing	A logical scalar: should a decreasing clipping window be used?
n	An integer value giving the number of iterations.

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

References

Morháč, M., Kliman, J., Matoušek, V., Veselský, M. & Turzo, I. (1997). Background elimination methods for multidimensional gamma-ray spectra. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 401(1), p. 113-132. doi:10.1016/S01689002(97)010231

Morháč, M. & Matoušek, V. (2008). Peak Clipping Algorithms for Background Estimation in Spectroscopic Data. *Applied Spectroscopy*, 62(1), p. 91-106. doi:10.1366/000370208783412762

Ryan, C. G., Clayton, E., Griffin, W. L., Sie, S. H. & 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 Section B: Beam Interactions with Materials and Atoms*, 34(3), p. 396-402. doi:10.1016/0168583X(88)900638

See Also

signal_correct()

Other baseline estimation methods: baseline_linear(), baseline_peakfilling(), baseline_polynomial(), baseline_rollingball(), baseline_rubberband()

Examples

```
## gamma-ray spectrometry
data("BEGe")

## Subset from 2.75 to 200 keV
BEGe <- signal_select(BEGe, from = 3, to = 200)

## Plot spectrum
plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")

## SNIP baseline
```

```
baseline <- baseline_snip(BEGe, LLS = FALSE, decreasing = FALSE, n = 100)

plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")
lines(baseline, type = "l", col = "red")
```

BEGe *Gamma-Ray Spectrometry*

Description

Gamma-Ray Spectrometry

Usage

```
BEGe
```

Format

A `data.frame` with 8192 rows (channels) and 2 variables.

energy (keV)

count

See Also

Other datasets: `LaBr`, `XRD`

Examples

```
data("BEGe")
plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")
```

`integrate_rectangle` *Rectangle Rule*

Description

Approximates the definite integral by using the rectangle rule.

Usage

```
integrate_rectangle(x, y, ...)

## S4 method for signature 'numeric,numeric'
integrate_rectangle(x, y, right = FALSE)

## S4 method for signature 'ANY,missing'
integrate_rectangle(x)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Currently not used.
right	A logical scalar: should the right rule be used instead of the left rule?

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other integration methods: integrate_trapezoid()

Examples

```
## Calculate the area under the sine curve from 0 to pi
# integrate(f = function(x) x^3, lower = 0, upper = 2)
x <- seq(0, 2, len = 101)
y <- x^3

plot(x, y, type = "l")

integrate_rectangle(x, y, right = FALSE) # 3.9204
integrate_rectangle(x, y, right = TRUE) # 4.0804
integrate_trapezoid(x, y) # 4.0004
```

---

integrate_trapezoid *Trapezoidal Rule*

Description

Approximates the definite integral by using the trapezoidal rule.

Usage

```
integrate_trapezoid(x, y, ...)

## S4 method for signature 'numeric,numeric'
integrate_trapezoid(x, y)

## S4 method for signature 'ANY,missing'
integrate_trapezoid(x)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Currently not used.

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other integration methods: integrate_rectangle()

Examples

```
## Calculate the area under the sine curve from 0 to pi
# integrate(f = function(x) x^3, lower = 0, upper = 2)
x <- seq(0, 2, len = 101)
y <- x^3

plot(x, y, type = "l")

integrate_rectangle(x, y, right = FALSE) # 3.9204
integrate_rectangle(x, y, right = TRUE) # 4.0804
integrate_trapezoid(x, y) # 4.0004
```

LaBr *Gamma-Ray Spectrometry*

Description

Gamma-Ray Spectrometry

Usage

```
LaBr
```

Format

A data.frame with 1024 rows (channels) and 2 variables.

energy (keV)

count

See Also

Other datasets: BEGe, XRD

Examples

```
data("LaBr")
plot(LaBr, type = "l", xlab = "Energy (keV)", ylab = "Count")
```

---

peaks_find *Find Peaks*

Description

Finds local maxima in sequential data.

Usage

```
peaks_find(x, y, ...)

## S4 method for signature 'numeric,numeric'
peaks_find(x, y, method = "MAD", SNR = 2, m = NULL, ...)

## S4 method for signature 'ANY,missing'
peaks_find(x, method = "MAD", SNR = 2, m = NULL, ...)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Extra parameters to be passed to internal methods.
method	A character string specifying the method to be used for background noise estimation (see below).
SNR	An integer giving the signal-to-noise-ratio for peak detection (see below).
m	An odd integer giving the window size (i.e. the number of adjacent points to be used). If NULL, 5% of the data points is used as the half window size.

Details

A local maximum has to be the highest one in the given window and has to be higher than $SNR \times noise$ to be recognized as peak.

The following methods are available for noise estimation:

MAD Median Absolute Deviation.

Note that to improve peak detection, it may be helpful to smooth the data and remove the baseline beforehand.

Value

Returns a list with two components x and y.

Note

There will be $(m-1)/2$ points both at the beginning and at the end of the data series for which a complete m-width window cannot be obtained. To prevent data loss, progressively wider/narrower windows are used at both ends of the data series.

Adapted from Stasia Grinberg's findPeaks function.

Author(s)

N. Frerebeau

See Also

Other peaks detection methods: peaks_fwhm()

Examples

```
## X-ray diffraction
data("XRD")

## Whittaker smoothing
smooth <- smooth_whittaker(XRD, lambda = 1000, d = 2, sparse = TRUE)

## 4S Peak Filling baseline
baseline <- baseline_peakfilling(smooth, n = 10, m = 5, by = 10)

plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")
lines(baseline, type = "l", col = "red")

## Correct baseline
XRD <- signal_drift(XRD, lag = baseline, subtract = TRUE)

## Find peaks
peaks <- peaks_find(XRD, SNR = 3, m = 11)

plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")
lines(peaks, type = "p", pch = 16, col = "red")
abline(h = attr(peaks, "noise"), lty = 2) # noise threshold

## Half-Width at Half-Maximum
x <- seq(-4, 4, length = 1000)
y <- dnorm(x)

peaks_fwhm(x, y, center = 0) # Expected: 2 * sqrt(2 * log(2))
```

peaks_fwhm *Half-Width at Half-Maximum*

Description

Estimates the Half-Width at Half-Maximum (FWHM) for a given peak.

Usage

```
peaks_fwhm(x, y, ...)

## S4 method for signature 'numeric,numeric'
peaks_fwhm(x, y, center)

## S4 method for signature 'ANY,missing'
peaks_fwhm(x, center)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Currently not used.
center	A numeric value giving the peak position in x units.

Details

It tries to get the smallest possible estimate.

Value

A numeric value.

Author(s)

N. Frerebeau

See Also

Other peaks detection methods: peaks_find()

Examples

```
## X-ray diffraction
data("XRD")

## Whittaker smoothing
smooth <- smooth_whittaker(XRD, lambda = 1000, d = 2, sparse = TRUE)
```

```
## 4S Peak Filling baseline
baseline <- baseline_peakfilling(smooth, n = 10, m = 5, by = 10)

plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")
lines(baseline, type = "l", col = "red")

## Correct baseline
XRD <- signal_drift(XRD, lag = baseline, subtract = TRUE)

## Find peaks
peaks <- peaks_find(XRD, SNR = 3, m = 11)

plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")
lines(peaks, type = "p", pch = 16, col = "red")
abline(h = attr(peaks, "noise"), lty = 2) # noise threshold

## Half-Width at Half-Maximum
x <- seq(-4, 4, length = 1000)
y <- dnorm(x)

peaks_fwhm(x, y, center = 0) # Expected: 2 * sqrt(2 * log(2))
```

`replace_negative` *Replace Negative Values*

Description

Replace Negative Values

Usage

```
replace_negative(x, y, ...)

## S4 method for signature 'numeric,numeric'
replace_negative(x, y, value = 0)

## S4 method for signature 'ANY,missing'
replace_negative(x, value = 0)
```

Arguments

`x, y`	A `numeric` vector. If `y` is missing, an attempt is made to interpret `x` in a suitable way (see `grDevices::xy.coords()`).
`...`	Extra parameters to be passed to `threshold`.
`value`	A `numeric` value to replace negative values.

Value

Returns a `list` with two components `x` and `y`.

Author(s)

N. Frerebeau

See Also

Other replacement methods: `replace_threshold()`

`replace_threshold` *Replace Values Below a Given Threshold*

Description

Replace Values Below a Given Threshold

Usage

```
replace_threshold(x, y, threshold, ...)

## S4 method for signature 'numeric,numeric,`function`'
replace_threshold(x, y, threshold, value = 0, ...)

## S4 method for signature 'ANY,missing,`function`'
replace_threshold(x, threshold, value = 0, ...)

## S4 method for signature 'numeric,numeric,numeric'
replace_threshold(x, y, threshold, value = 0, ...)

## S4 method for signature 'ANY,missing,numeric'
replace_threshold(x, threshold, value = 0, ...)
```

Arguments

`x, y`	A `numeric` vector. If `y` is missing, an attempt is made to interpret `x` in a suitable way (see `grDevices::xy.coords()`).
`threshold`	A `numeric` value or a `function` that takes a `numeric` vector as argument and returns a single `numeric` value.
`...`	Extra parameters to be passed to `threshold`.
`value`	A `numeric` value to replace values below `threshold`.

Value

Returns a `list` with two components `x` and `y`.

Author(s)

N. Frerebeau

See Also

Other replacement methods: `replace_negative()`

`resample_bin` *Bin*

Description

Averages `x` values and applies a function to the corresponding `y` values.

Usage

```
resample_bin(x, y, ...)

## S4 method for signature 'numeric,numeric'
resample_bin(x, y, by, f = mean, ...)

## S4 method for signature 'ANY,missing'
resample_bin(x, y, by, f = sum)
```

Arguments

`x, y`	A `numeric` vector. If `y` is missing, an attempt is made to interpret `x` in a suitable way (see `grDevices::xy.coords()`).
`...`	Extra parameters to be passed to `f`.
`by`	An `integer` specifying the binning ratio (i.e. the number of points to be grouped together; see `window_tumbling()`).
`f`	A `function` that takes a `numeric` vector of intensities as argument and returns a single `numeric` vector. Used to estimate the local representative value in each bin (defaults to `sum()`; see examples).

Value

Returns a `list` with two components `x` and `y`.

Author(s)

N. Frerebeau

See Also

Other resampling methods: `resample_down()`, `resample_interpolate()`

Examples

```
## X-ray diffraction
data("XRD")

## Plot spectrum
plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")

## Bin by 3
XRD_bin_mean <- resample_bin(XRD, by = 3, f = mean)
XRD_bin_min <- resample_bin(XRD, by = 3, f = min)

plot(XRD, type = "l", xlim = c(25, 35),
     xlab = expression(2*theta), ylab = "Count")
lines(XRD_bin_mean, type = "l", col = "red")
lines(XRD_bin_min, type = "l", col = "green")

## Downsample by 10
XRD_down <- resample_down(XRD, by = 10)

plot(XRD, type = "l", xlim = c(20, 40),
     xlab = expression(2*theta), ylab = "Count")
lines(XRD_down, type = "l", col = "red")

## Linearly interpolate
XRD_approx <- resample_interpolate(XRD, from = 20, to = 40, by = 0.02)

plot(XRD, type = "l", xlim = c(20, 40),
     xlab = expression(2*theta), ylab = "Count")
lines(XRD_approx, type = "l", col = "red")
```

`resample_down` *Downsample*

Description

Downsample

Usage

```
resample_down(x, y, ...)

## S4 method for signature 'numeric,numeric'
resample_down(x, y, by)

## S4 method for signature 'ANY,missing'
resample_down(x, y, by)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Currently not used.
by	An integer specifying the downsampling factor.

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other resampling methods: resample_bin(), resample_interpolate()

Examples

```
## X-ray diffraction
data("XRD")

## Plot spectrum
plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")

## Bin by 3
XRD_bin_mean <- resample_bin(XRD, by = 3, f = mean)
XRD_bin_min <- resample_bin(XRD, by = 3, f = min)

plot(XRD, type = "l", xlim = c(25, 35),
     xlab = expression(2*theta), ylab = "Count")
lines(XRD_bin_mean, type = "l", col = "red")
lines(XRD_bin_min, type = "l", col = "green")

## Downsample by 10
XRD_down <- resample_down(XRD, by = 10)

plot(XRD, type = "l", xlim = c(20, 40),
     xlab = expression(2*theta), ylab = "Count")
lines(XRD_down, type = "l", col = "red")

## Linearly interpolate
XRD_approx <- resample_interpolate(XRD, from = 20, to = 40, by = 0.02)

plot(XRD, type = "l", xlim = c(20, 40),
     xlab = expression(2*theta), ylab = "Count")
lines(XRD_approx, type = "l", col = "red")
```

`resample_interpolate` *Linearly Interpolate*

Description

Linearly Interpolate

Usage

```
resample_interpolate(x, y, ...)

## S4 method for signature 'numeric,numeric'
resample_interpolate(x, y, from, to, by, ...)

## S4 method for signature 'ANY,missing'
resample_interpolate(x, y, from, to, by, ...)
```

Arguments

`x, y`	A `numeric` vector. If y is missing, an attempt is made to interpret x in a suitable way (see `grDevices::xy.coords()`).
`...`	Extra arguments to be passed to `stats::approx()`.
`from`	A length-one `numeric` vector giving the starting value of the sequence where interpolation is to take place.
`to`	A length-one `numeric` vector giving the end value of the sequence where interpolation is to take place.
`by`	A length-one `numeric` vector specifying the increment of the sequence.

Value

Returns a `list` with two components x and y.

Author(s)

N. Frerebeau

See Also

Other resampling methods: `resample_bin()`, `resample_down()`

Examples

```
## X-ray diffraction
data("XRD")

## Plot spectrum
plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")
```

```
## Bin by 3
XRD_bin_mean <- resample_bin(XRD, by = 3, f = mean)
XRD_bin_min <- resample_bin(XRD, by = 3, f = min)

plot(XRD, type = "l", xlim = c(25, 35),
     xlab = expression(2*theta), ylab = "Count")
lines(XRD_bin_mean, type = "l", col = "red")
lines(XRD_bin_min, type = "l", col = "green")

## Downsample by 10
XRD_down <- resample_down(XRD, by = 10)

plot(XRD, type = "l", xlim = c(20, 40),
     xlab = expression(2*theta), ylab = "Count")
lines(XRD_down, type = "l", col = "red")

## Linearly interpolate
XRD_approx <- resample_interpolate(XRD, from = 20, to = 40, by = 0.02)

plot(XRD, type = "l", xlim = c(20, 40),
     xlab = expression(2*theta), ylab = "Count")
lines(XRD_approx, type = "l", col = "red")
```

`rescale_area` *Normalize intensities by AUC*

Description

Rescales intensities so that the area under the curve (AUC) is equal to 1.

Usage

```
rescale_area(x, y, ...)

## S4 method for signature 'numeric,numeric'
rescale_area(x, y, method = c("rectangle", "trapezoid"), ...)

## S4 method for signature 'ANY,missing'
rescale_area(x, method = c("rectangle", "trapezoid"), ...)
```

Arguments

`x, y`	A `numeric` vector. If `y` is missing, an attempt is made to interpret `x` in a suitable way (see `grDevices::xy.coords()`).
`...`	Currently not used.
`method`	A `character` string specifying the method for integration. It must be one of `"rectangle"` or `"trapezoid"`. Any unambiguous substring can be given.

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other normalization methods: rescale_range(), rescale_total(), rescale_transform()

Examples

```
## gamma-ray spectrometry
data("BEGe")

## Subset from 2.75 to 200 keV
BEGe <- signal_select(BEGe, from = 3, to = 200)

## Plot spectrum
plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")

## Normalize by area under the curve
BEGe_area <- rescale_area(BEGe)
plot(BEGe_area, type = "l", xlab = "Energy (keV)", ylab = "Count")
integrate_rectangle(BEGe)
integrate_rectangle(BEGe_area)

## Rescale so that intensities sum to 1
BEGe_total <- rescale_total(BEGe, total = 1)
plot(BEGe_total, type = "l", xlab = "Energy (keV)", ylab = "Count")

## Rescale intensities to 0-1
BEGe_range <- rescale_range(BEGe, min = 0, max = 1)
plot(BEGe_range, type = "l", xlab = "Energy (keV)", ylab = "Count")
```

---

rescale_range *Rescales intensities to have specified minimum and maximum*

Description

Rescales intensities to have specified minimum and maximum.

Usage

```
rescale_range(x, y, ...)

rescale_min(x, y, ...)
```

```
rescale_max(x, y, ...)

## S4 method for signature 'numeric,numeric'
rescale_range(x, y, min = 0, max = 1)

## S4 method for signature 'ANY,missing'
rescale_range(x, min = 0, max = 1)

## S4 method for signature 'numeric,numeric'
rescale_min(x, y, min = 0)

## S4 method for signature 'ANY,missing'
rescale_min(x, min = 0)

## S4 method for signature 'numeric,numeric'
rescale_max(x, y, max = 1)

## S4 method for signature 'ANY,missing'
rescale_max(x, max = 1)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Currently not used.
min	A legnth-one numeric vector specifying the output minimum.
max	A legnth-one numeric vector specifying the output maximum.

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other normalization methods: rescale_area(), rescale_total(), rescale_transform()

Examples

```
## gamma-ray spectrometry
data("BEGe")

## Subset from 2.75 to 200 keV
BEGe <- signal_select(BEGe, from = 3, to = 200)

## Plot spectrum
```

```
plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")

## Normalize by area under the curve
BEGe_area <- rescale_area(BEGe)
plot(BEGe_area, type = "l", xlab = "Energy (keV)", ylab = "Count")
integrate_rectangle(BEGe)
integrate_rectangle(BEGe_area)

## Rescale so that intensities sum to 1
BEGe_total <- rescale_total(BEGe, total = 1)
plot(BEGe_total, type = "l", xlab = "Energy (keV)", ylab = "Count")

## Rescale intensities to 0-1
BEGe_range <- rescale_range(BEGe, min = 0, max = 1)
plot(BEGe_range, type = "l", xlab = "Energy (keV)", ylab = "Count")
```

`rescale_total` *Rescale intensities to sum to a specified value*

Description

Rescales intensities to sum to a specified value.

Usage

```
rescale_total(x, y, ...)

## S4 method for signature 'numeric,numeric'
rescale_total(x, y, total = 1)

## S4 method for signature 'ANY,missing'
rescale_total(x, total = 1)
```

Arguments

`x, y`	A `numeric` vector. If `y` is missing, an attempt is made to interpret `x` in a suitable way (see `grDevices::xy.coords()`).
`...`	Currently not used.
`total`	A legnth-one `numeric` vector specifying the output total. Defaults to 1, i.e. normalizes by total intensity.

Value

Returns a `list` with two components `x` and `y`.

Author(s)

N. Frerebeau

See Also

Other normalization methods: rescale_area(), rescale_range(), rescale_transform()

Examples

```
## gamma-ray spectrometry
data("BEGe")

## Subset from 2.75 to 200 keV
BEGe <- signal_select(BEGe, from = 3, to = 200)

## Plot spectrum
plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")

## Normalize by area under the curve
BEGe_area <- rescale_area(BEGe)
plot(BEGe_area, type = "l", xlab = "Energy (keV)", ylab = "Count")
integrate_rectangle(BEGe)
integrate_rectangle(BEGe_area)

## Rescale so that intensities sum to 1
BEGe_total <- rescale_total(BEGe, total = 1)
plot(BEGe_total, type = "l", xlab = "Energy (keV)", ylab = "Count")

## Rescale intensities to 0-1
BEGe_range <- rescale_range(BEGe, min = 0, max = 1)
plot(BEGe_range, type = "l", xlab = "Energy (keV)", ylab = "Count")
```

---

rescale_transform *Transform Intensities*

Description

Transform Intensities

Usage

```
rescale_transform(x, y, ...)

## S4 method for signature 'numeric,numeric'
rescale_transform(x, y, f, ...)

## S4 method for signature 'ANY,missing'
rescale_transform(x, f, ...)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Extra arguments to be passed to f.
f	A function that takes a numeric vector of intensities as argument and returns a numeric vector.

Details

Transformation of intensities can be used to improve the identification of peaks with a low signal-to-noise ratio.

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other normalization methods: rescale_area(), rescale_range(), rescale_total()

Examples

```
## gamma-ray spectrometry
data("BEGe")

## Subset from 2.75 to 200 keV
BEGe <- signal_select(BEGe, from = 3, to = 200)

## Plot spectrum
plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")

## Transform intensities
BEGe_trans <- rescale_transform(BEGe, f = sqrt)
plot(BEGe_trans, type = "l", xlab = "Energy (keV)", ylab = "sqrt(Count)")
```

---

signal_bind *Bind*

Description

Combines XY objects.

Usage

```
signal_bind(...)

## S4 method for signature 'ANY'
signal_bind(...)
```

Arguments

...	Any object that can be interpreted in a suitable way (see grDevices::xy.coords()).

Value

Returns a matrix of intensities.

Author(s)

N. Frerebeau

See Also

Other signal processing methods: signal_correct(), signal_drift(), signal_mean(), signal_shift(), subset()

Examples

```
## X-ray diffraction
data("XRD")

XRD1 <- signal_drift(XRD, lag = 1500)

## Bind
XRD_bind <- signal_bind(XRD, XRD1)
XRD_bind[, 1:10]

## Mean
XRD_mean <- signal_mean(XRD, XRD1)

plot(NULL, type = "l", xlim = c(10, 70) , ylim = c(3000, 36000),
     xlab = expression(2*theta), ylab = "Count")
lines(XRD, type = "l")
lines(XRD1, type = "l")
lines(XRD_mean, type = "l", col = "red")
```

signal_correct *Baseline Correction*

Description

Baseline Correction

Usage

```
signal_correct(x, y, ...)

## S4 method for signature 'numeric,numeric'
signal_correct(x, y, method = c("linear", "rubberband", "SNIP", "4S"), ...)

## S4 method for signature 'ANY,missing'
signal_correct(x, method = c("linear", "rubberband", "SNIP", "4S"), ...)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Extra arguments to be passed to baseline_*() (see details).
method	A character string specifying the method for baseline estimation. It must be one of "linear", "rubberband", "SNIP" or "4S" (see details). Any unambiguous substring can be given.

Details

Available methods for baseline estimation:

linear Linear baseline estimation (see baseline_linear()).

rubberband Rubberband baseline estimation (see baseline_rubberband()).

SNIP Sensitive Nonlinear Iterative Peak clipping algorithm (see baseline_snip()).

4S 4S Peak Filling (see baseline_peakfilling()).

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other signal processing methods: signal_bind(), signal_drift(), signal_mean(), signal_shift(), subset()

Examples

```
## gamma-ray spectrometry
data("BEGe")

## Subset from 2.75 to 200 keV
BEGe <- signal_select(BEGe, from = 3, to = 200)

## Drift
baseline <- baseline_snip(BEGe)
BEGe_drif <- signal_drift(BEGe, lag = baseline, subtract = TRUE)

plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")
lines(BEGe_drif, type = "l", col = "red")

## Correct
BEGe_corr <- signal_correct(BEGe, method = "SNIP")

plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")
lines(BEGe_corr, type = "l", col = "red")
```

`signal_drift` *Drift Intensities*

Description

Drift Intensities

Usage

```
signal_drift(x, y, lag, ...)

## S4 method for signature 'numeric,numeric,numeric'
signal_drift(x, y, lag)

## S4 method for signature 'ANY,missing,ANY'
signal_drift(x, lag, subtract = FALSE)
```

Arguments

`x, y`	A `numeric` vector. If `y` is missing, an attempt is made to interpret `x` in a suitable way (see `grDevices::xy.coords()`).
`lag`	A `numeric` vector specifying the offset or any object that can be interpreted in a suitable way (see `grDevices::xy.coords()`)
`...`	Currently not used.
`subtract`	A `logical` scalar: should `lag` be subtracted to `y`?

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other signal processing methods: signal_bind(), signal_correct(), signal_mean(), signal_shift(), subset()

Examples

```
## gamma-ray spectrometry
data("BEGe")

## Subset from 2.75 to 200 keV
BEGe <- signal_select(BEGe, from = 3, to = 200)

## Drift
BEGe_plus <- signal_drift(BEGe, lag = 250)
BEGe_minus <- signal_drift(BEGe, lag = 250, subtract = TRUE)

plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")
lines(BEGe_plus, type = "l", col = "red")
lines(BEGe_minus, type = "l", col = "green")
```

---

signal_mean *Mean Intensities*

Description

Mean Intensities

Usage

```
signal_mean(...)

## S4 method for signature 'ANY'
signal_mean(...)
```

Arguments

...	Any object that can be interpreted in a suitable way (see grDevices::xy.coords()).

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other signal processing methods: signal_bind(), signal_correct(), signal_drift(), signal_shift(), subset()

Examples

```
## X-ray diffraction
data("XRD")

XRD1 <- signal_drift(XRD, lag = 1500)

## Bind
XRD_bind <- signal_bind(XRD, XRD1)
XRD_bind[, 1:10]

## Mean
XRD_mean <- signal_mean(XRD, XRD1)

plot(NULL, type = "l", xlim = c(10, 70) , ylim = c(3000, 36000),
     xlab = expression(2*theta), ylab = "Count")
lines(XRD, type = "l")
lines(XRD1, type = "l")
lines(XRD_mean, type = "l", col = "red")
```

---

signal_shift *Shift the X Scale*

Description

Shifts the x scale by a given value.

Usage

```
signal_shift(x, y, lag, ...)

## S4 method for signature 'numeric,numeric'
signal_shift(x, y, lag)

## S4 method for signature 'ANY,missing'
signal_shift(x, lag)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
lag	A numeric vector specifying the offset.
...	Currently not used.

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other signal processing methods: signal_bind(), signal_correct(), signal_drift(), signal_mean(), subset()

Examples

```
## X-ray diffraction
data("XRD")

## Plot spectrum
plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")

## Shift by one degree
XRD_offset <- signal_shift(XRD, lag = 1)
lines(XRD_offset, type = "l", col = "red")
```

---

smooth_loess *Loess Smoothing*

Description

Smoothes intensities by loess fitting.

Usage

```
smooth_loess(x, y, ...)

## S4 method for signature 'numeric,numeric'
smooth_loess(x, y, span = 0.75, ...)

## S4 method for signature 'ANY,missing'
smooth_loess(x, span = 0.75, ...)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Extra arguments to be passed to stats::loess().
span	An integer specifying the degree of smoothing (see stats::loess()).

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other smoothing methods: smooth_rectangular(), smooth_savitzky(), smooth_triangular(), smooth_whittaker()

Examples

```
## Simulate data with some noise
x <- seq(-4, 4, length = 100)
y <- dnorm(x) + rnorm(100, mean = 0, sd = 0.01)

## Plot spectrum
plot(x, y, type = "l", xlab = "", ylab = "")

## Rectangular smoothing
unweighted <- smooth_rectangular(x, y, m = 3)
plot(unweighted, type = "l", xlab = "", ylab = "")

## Triangular smoothing
weighted <- smooth_triangular(x, y, m = 5)
plot(weighted, type = "l", xlab = "", ylab = "")

## Loess smoothing
loess <- smooth_loess(x, y, span = 0.75)
plot(loess, type = "l", xlab = "", ylab = "")

## SavitzkyGolay filter
savitzky <- smooth_savitzky(x, y, m = 21, p = 2)
plot(savitzky, type = "l", xlab = "", ylab = "")

## Whittaker smoothing
whittaker <- smooth_whittaker(x, y, lambda = 1600, d = 2)
plot(whittaker, type = "l", xlab = "", ylab = "")
```

`smooth_rectangular` *Rectangular Smoothing*

Description

Unweighted sliding-average or rectangular Smoothing.

Usage

```
smooth_rectangular(x, y, ...)

## S4 method for signature 'numeric,numeric'
smooth_rectangular(x, y, m = 3)

## S4 method for signature 'ANY,missing'
smooth_rectangular(x, m)
```

Arguments

`x, y`	A `numeric` vector. If `y` is missing, an attempt is made to interpret `x` in a suitable way (see `grDevices::xy.coords()`).
`...`	Currently not used.
`m`	An odd `integer` giving the window size (i.e. the number of adjacent points to be used; see `window_sliding()`).

Details

It replaces each point in the signal with the average of m adjacent points.

Value

Returns a `list` with two components `x` and `y`.

Note

There will be $(m-1)/2$ points both at the beginning and at the end of the data series for which a complete m-width window cannot be obtained. To prevent data loss, progressively wider/narrower windows are used at both ends of the data series.

Author(s)

N. Frerebeau

See Also

Other smoothing methods: `smooth_loess()`, `smooth_savitzky()`, `smooth_triangular()`, `smooth_whittaker()`

Examples

```
## Simulate data with some noise
x <- seq(-4, 4, length = 100)
y <- dnorm(x) + rnorm(100, mean = 0, sd = 0.01)

## Plot spectrum
plot(x, y, type = "l", xlab = "", ylab = "")

## Rectangular smoothing
unweighted <- smooth_rectangular(x, y, m = 3)
plot(unweighted, type = "l", xlab = "", ylab = "")

## Triangular smoothing
weighted <- smooth_triangular(x, y, m = 5)
plot(weighted, type = "l", xlab = "", ylab = "")

## Loess smoothing
loess <- smooth_loess(x, y, span = 0.75)
plot(loess, type = "l", xlab = "", ylab = "")

## SavitzkyGolay filter
savitzky <- smooth_savitzky(x, y, m = 21, p = 2)
plot(savitzky, type = "l", xlab = "", ylab = "")

## Whittaker smoothing
whittaker <- smooth_whittaker(x, y, lambda = 1600, d = 2)
plot(whittaker, type = "l", xlab = "", ylab = "")
```

---

smooth_savitzky *Savitzky-Golay Filter*

Description

Savitzky-Golay Filter

Usage

```
smooth_savitzky(x, y, ...)

## S4 method for signature 'numeric,numeric'
smooth_savitzky(x, y, m = 3, p = 2)

## S4 method for signature 'ANY,missing'
smooth_savitzky(x, m, p)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).

...	Currently not used.
m	An odd integer giving the window size (i.e. the number of adjacent points to be used).
p	An integer giving the degree of the polynomial to be used.

Details

This method is based on the least-squares fitting of polynomials to segments of m adjacent points.

Value

Returns a list with two components x and y.

Note

There will be $(m-1)/2$ points both at the beginning and at the end of the data series for which a complete m-width window cannot be obtained. To prevent data loss, the original $(m-1)/2$ points at both ends of the data series are preserved.

Author(s)

N. Frerebeau

References

Gorry, P. A. (1990). General Least-Squares Smoothing and Differentiation by the Convolution (Savitzky-Golay) Method. *Analytical Chemistry*, 62(6), p. 570-573. doi:10.1021/ac00205a007.

Savitzky, A. & Golay, M. J. E. (1964). Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, 36(8), p. 1627-1639. doi:10.1021/ac60214a047.

See Also

Other smoothing methods: smooth_loess(), smooth_rectangular(), smooth_triangular(), smooth_whittaker()

Examples

```
## Simulate data with some noise
x <- seq(-4, 4, length = 100)
y <- dnorm(x) + rnorm(100, mean = 0, sd = 0.01)

## Plot spectrum
plot(x, y, type = "l", xlab = "", ylab = "")

## Rectangular smoothing
unweighted <- smooth_rectangular(x, y, m = 3)
plot(unweighted, type = "l", xlab = "", ylab = "")

## Triangular smoothing
weighted <- smooth_triangular(x, y, m = 5)
```

```
plot(weighted, type = "l", xlab = "", ylab = "")

## Loess smoothing
loess <- smooth_loess(x, y, span = 0.75)
plot(loess, type = "l", xlab = "", ylab = "")

## SavitzkyGolay filter
savitzky <- smooth_savitzky(x, y, m = 21, p = 2)
plot(savitzky, type = "l", xlab = "", ylab = "")

## Whittaker smoothing
whittaker <- smooth_whittaker(x, y, lambda = 1600, d = 2)
plot(whittaker, type = "l", xlab = "", ylab = "")
```

`smooth_triangular` *Triangular Smoothing*

Description

Weighted sliding-average or triangular smoothing.

Usage

```
smooth_triangular(x, y, ...)

## S4 method for signature 'numeric,numeric'
smooth_triangular(x, y, m = 3)

## S4 method for signature 'ANY,missing'
smooth_triangular(x, m)
```

Arguments

`x, y`	A `numeric` vector. If `y` is missing, an attempt is made to interpret `x` in a suitable way (see `grDevices::xy.coords()`).
`...`	Currently not used.
`m`	An odd `integer` giving the window size (i.e. the number of adjacent points to be used; see `window_sliding()`).

Details

It replaces each point in the signal with the weighted mean of m adjacent points.

Value

Returns a `list` with two components `x` and `y`.

Note

There will be $(m-1)/2$ points both at the beginning and at the end of the data series for which a complete m-width window cannot be obtained. To prevent data loss, progressively wider/narrower windows are used at both ends of the data series.

Author(s)

N. Frerebeau

See Also

Other smoothing methods: `smooth_loess()`, `smooth_rectangular()`, `smooth_savitzky()`, `smooth_whittaker()`

Examples

```
## Simulate data with some noise
x <- seq(-4, 4, length = 100)
y <- dnorm(x) + rnorm(100, mean = 0, sd = 0.01)

## Plot spectrum
plot(x, y, type = "l", xlab = "", ylab = "")

## Rectangular smoothing
unweighted <- smooth_rectangular(x, y, m = 3)
plot(unweighted, type = "l", xlab = "", ylab = "")

## Triangular smoothing
weighted <- smooth_triangular(x, y, m = 5)
plot(weighted, type = "l", xlab = "", ylab = "")

## Loess smoothing
loess <- smooth_loess(x, y, span = 0.75)
plot(loess, type = "l", xlab = "", ylab = "")

## SavitzkyGolay filter
savitzky <- smooth_savitzky(x, y, m = 21, p = 2)
plot(savitzky, type = "l", xlab = "", ylab = "")

## Whittaker smoothing
whittaker <- smooth_whittaker(x, y, lambda = 1600, d = 2)
plot(whittaker, type = "l", xlab = "", ylab = "")
```

`smooth_whittaker` *Whittaker Smoothing*

Description

Whittaker Smoothing

Usage

```
smooth_whittaker(x, y, ...)

## S4 method for signature 'numeric,numeric'
smooth_whittaker(x, y, lambda = 1600, d = 2, sparse = FALSE)

## S4 method for signature 'ANY,missing'
smooth_whittaker(x, lambda = 1600, d = 2, sparse = FALSE)
```

Arguments

x, y	A numeric vector. If y is missing, an attempt is made to interpret x in a suitable way (see grDevices::xy.coords()).
...	Currently not used.
lambda	An integer giving the smoothing parameter. The larger lambda is, the smoother the curve.
d	An integer specifying the order of the penalty.
sparse	A logical scalar: should sparse matrices be used for computation? If TRUE, **Matrix** is required.

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

References

Eilers, P. H. C. (2003). A Perfect Smoother. *Analytical Chemistry*, 75(14): 3631-36. doi:10.1021/ac034173t.

See Also

Other smoothing methods: smooth_loess(), smooth_rectangular(), smooth_savitzky(), smooth_triangular()

Examples

```
## Simulate data with some noise
x <- seq(-4, 4, length = 100)
y <- dnorm(x) + rnorm(100, mean = 0, sd = 0.01)

## Plot spectrum
plot(x, y, type = "l", xlab = "", ylab = "")

## Rectangular smoothing
unweighted <- smooth_rectangular(x, y, m = 3)
plot(unweighted, type = "l", xlab = "", ylab = "")
```

```
## Triangular smoothing
weighted <- smooth_triangular(x, y, m = 5)
plot(weighted, type = "l", xlab = "", ylab = "")

## Loess smoothing
loess <- smooth_loess(x, y, span = 0.75)
plot(loess, type = "l", xlab = "", ylab = "")

## SavitzkyGolay filter
savitzky <- smooth_savitzky(x, y, m = 21, p = 2)
plot(savitzky, type = "l", xlab = "", ylab = "")

## Whittaker smoothing
whittaker <- smooth_whittaker(x, y, lambda = 1600, d = 2)
plot(whittaker, type = "l", xlab = "", ylab = "")
```

`subset` *Subset*

Description

- `signal_select()` allows to subset by values of `x`.
- `signal_slice()` allows to subset by position along `x`.

Usage

```
signal_select(x, y, ...)

signal_slice(x, y, ...)

## S4 method for signature 'numeric,numeric'
signal_select(x, y, from, to)

## S4 method for signature 'ANY,missing'
signal_select(x, from, to)

## S4 method for signature 'numeric,numeric'
signal_slice(x, y, subset)

## S4 method for signature 'ANY,missing'
signal_slice(x, subset)
```

Arguments

`x, y`	A `numeric` vector. If `y` is missing, an attempt is made to interpret `x` in a suitable way (see `grDevices::xy.coords()`).
`...`	Currently not used.

from, to	A numeric value giving the first and last value (in x unit) to be selected.
subset	An integer vector giving either positive values to keep, or negative values to drop. The values provided must be either all positive or all negative (coerced to integer as by as.integer()).

Value

Returns a list with two components x and y.

Author(s)

N. Frerebeau

See Also

Other signal processing methods: signal_bind(), signal_correct(), signal_drift(), signal_mean(), signal_shift()

Examples

```
## gamma-ray spectrometry
data("BEGe")

## Plot spectrum
plot(BEGe, type = "l", xlab = "Energy (keV)", ylab = "Count")

## Subset from 2.75 keV to 200 keV
BEGe_1 <- signal_select(BEGe, from = 3, to = 200)

## Plot spectrum
plot(BEGe_1, type = "l", xlab = "Energy (keV)", ylab = "Count")

## Subset from the 20th to the 1250th value
BEGe_2 <- signal_slice(BEGe, subset = 20:1250)

## Plot spectrum
plot(BEGe_2, type = "l", xlab = "Energy (keV)", ylab = "Count")
```

---

window_sliding *Sliding Windows*

Description

There will be $(m - 1)/2$ points both at the beginning and at the end of the data series for which a complete m-width window cannot be obtained. To prevent data loss, progressively wider/narrower windows are evaluated at both ends of the data series.

Usage

```
window_sliding(n, m, ...)

## S4 method for signature 'integer,integer'
window_sliding(n, m, i = NULL)

## S4 method for signature 'numeric,numeric'
window_sliding(n, m, i = NULL)
```

Arguments

n	An integer giving the length of the data series (will be coerced with as.integer() and hence truncated toward zero).
m	An odd integer giving the window size, i.e. the number of adjacent points to be used (will be coerced with as.integer() and hence truncated toward zero).
...	Currently not used.
i	A vector integer specifying the indices of the data points for which windows should be returned. If NULL (the default), windows are evaluated for each data point.

Value

Returns a length-n list of integer vectors (indices of the data points in each window).

Author(s)

N. Frerebeau

See Also

Other moving windows: window_tumbling()

Examples

```
## Length of the data series
n <- 10

## Progressive sliding windows
sliding <- window_sliding(n = n, m = 5)

plot(NULL, xlim = c(1, n), ylim = c(1, 10.5), xlab = "Index", ylab = "Window")
for (i in seq_along(sliding)) {
  w <- sliding[[i]]
  text(x = w, y = rep(i, length(w)), labels = w, pos = 3)
  lines(w, rep(i, length(w)), type = "l", lwd = 2)
}

## Tumbling windows
## (compare with drop = TRUE)
```

```
tumbling <- window_tumbling(n = n, m = 3, drop = FALSE)

plot(NULL, xlim = c(1, n), ylim = c(1, 5.5), xlab = "Index", ylab = "Window")
for (i in seq_along(tumbling)) {
  w <- tumbling[[i]]
  text(x = w, y = rep(i, length(w)), labels = w, pos = 3)
  lines(w, rep(i, length(w)), type = "l", lwd = 2)
}
```

`window_tumbling` *Tumbling Windows*

Description

Tumbling Windows

Usage

```
window_tumbling(n, m, ...)

## S4 method for signature 'integer,integer'
window_tumbling(n, m, drop = FALSE)

## S4 method for signature 'numeric,numeric'
window_tumbling(n, m, drop = FALSE)
```

Arguments

`n`	An `integer` giving the length of the data series (will be coerced with `as.integer()` and hence truncated toward zero).
`m`	An `integer` giving the window size, i.e. the number of adjacent points to be used (will be coerced with `as.integer()` and hence truncated toward zero).
`...`	Currently not used.
`drop`	A `logical` scalar: if `m` is not a multiple of `n`, should the last data points be removed so that all windows have the same length?

Value

Returns a `list` of `integer` vectors (indices of the data points in each window).

Author(s)

N. Frerebeau

See Also

Other moving windows: `window_sliding()`

Examples

```
## Length of the data series
n <- 10

## Progressive sliding windows
sliding <- window_sliding(n = n, m = 5)

plot(NULL, xlim = c(1, n), ylim = c(1, 10.5), xlab = "Index", ylab = "Window")
for (i in seq_along(sliding)) {
  w <- sliding[[i]]
  text(x = w, y = rep(i, length(w)), labels = w, pos = 3)
  lines(w, rep(i, length(w)), type = "l", lwd = 2)
}

## Tumbling windows
## (compare with drop = TRUE)
tumbling <- window_tumbling(n = n, m = 3, drop = FALSE)

plot(NULL, xlim = c(1, n), ylim = c(1, 5.5), xlab = "Index", ylab = "Window")
for (i in seq_along(tumbling)) {
  w <- tumbling[[i]]
  text(x = w, y = rep(i, length(w)), labels = w, pos = 3)
  lines(w, rep(i, length(w)), type = "l", lwd = 2)
}
```

XRD *Powder X-ray Diffraction*

Description

Powder X-ray Diffraction

Usage

```
XRD
```

Format

A data.frame with 2989 rows and 2 variables.

theta

count

See Also

Other datasets: BEGe, LaBr

Examples

```
data("XRD")
plot(XRD, type = "l", xlab = expression(2*theta), ylab = "Count")
```

Index

∗ **baseline estimation methods**
baseline_linear, 3
baseline_peakfilling, 4
baseline_polynomial, 5
baseline_rollingball, 6
baseline_rubberband, 8
baseline_snip, 9
∗ **datasets**
BEGe, 11
LaBr, 13
XRD, 46
∗ **integration methods**
integrate_rectangle, 11
integrate_trapezoid, 12
∗ **moving windows**
window_sliding, 43
window_tumbling, 45
∗ **normalization methods**
rescale_area, 23
rescale_range, 24
rescale_total, 26
rescale_transform, 27
∗ **peaks detection methods**
peaks_find, 14
peaks_fwhm, 16
∗ **replacement methods**
replace_negative, 17
replace_threshold, 18
∗ **resampling methods**
resample_bin, 19
resample_down, 20
resample_interpolate, 22
∗ **signal processing methods**
signal_bind, 28
signal_correct, 30
signal_drift, 31
signal_mean, 32
signal_shift, 33
subset, 42
∗ **smoothing methods**
smooth_loess, 34
smooth_rectangular, 36
smooth_savitzky, 37
smooth_triangular, 39
smooth_whittaker, 40

as.integer(), *43–45*

baseline_linear, 3, *5–7*, *9, 10*
baseline_linear(), *30*
baseline_linear,ANY,missing-method
*(*baseline_linear*)*, 3
baseline_linear,numeric,numeric-method
*(*baseline_linear*)*, 3
baseline_linear-method
*(*baseline_linear*)*, 3
baseline_peakfilling, *3*, 4, *6, 7*, *9, 10*
baseline_peakfilling(), *30*
baseline_peakfilling,ANY,missing-method
*(*baseline_peakfilling*)*, 4
baseline_peakfilling,numeric,numeric-method
*(*baseline_peakfilling*)*, 4
baseline_peakfilling-method
*(*baseline_peakfilling*)*, 4
baseline_polynomial, *3*, *5*, 5, *7*, *9, 10*
baseline_polynomial,ANY,missing-method
*(*baseline_polynomial*)*, 5
baseline_polynomial,numeric,numeric-method
*(*baseline_polynomial*)*, 5
baseline_polynomial-method
*(*baseline_polynomial*)*, 5
baseline_rollingball, *3*, *5, 6*, 6, *9, 10*
baseline_rollingball,ANY,missing-method
*(*baseline_rollingball*)*, 6
baseline_rollingball,numeric,numeric-method
*(*baseline_rollingball*)*, 6
baseline_rollingball-method
*(*baseline_rollingball*)*, 6
baseline_rubberband, *3*, *5–7*, 8, *10*

baseline_rubberband(), *30*
baseline_rubberband,ANY,missing-method (baseline_rubberband), 8
baseline_rubberband,numeric,numeric-method (baseline_rubberband), 8
baseline_rubberband-method (baseline_rubberband), 8
baseline_snip, *3*, *5–7*, *9*, 9
baseline_snip(), *30*
baseline_snip,ANY,missing-method (baseline_snip), 9
baseline_snip,numeric,numeric-method (baseline_snip), 9
baseline_snip-method (baseline_snip), 9
BEGe, 11, *14*, *46*

character, *14*, *23*, *30*

data.frame, *11*, *13*, *46*

function, *18, 19*, *28*

grDevices::xy.coords(), *3–5*, *7, 8*, *10*, *12–14*, *16–19*, *21–23*, *25, 26*, *28–32*, *34–37*, *39*, *41, 42*

hyperSpec::spc.rubberband(), *9*

integer, *4, 5*, *7*, *10*, *14*, *19*, *21*, *35, 36*, *38, 39*, *41*, *43–45*
integrate_rectangle, 11, *13*
integrate_rectangle,ANY,missing-method (integrate_rectangle), 11
integrate_rectangle,numeric,numeric-method (integrate_rectangle), 11
integrate_rectangle-method (integrate_rectangle), 11
integrate_trapezoid, *12*, 12
integrate_trapezoid,ANY,missing-method (integrate_trapezoid), 12
integrate_trapezoid,numeric,numeric-method (integrate_trapezoid), 12
integrate_trapezoid-method (integrate_trapezoid), 12

LaBr, *11*, 13, *46*
list, *3*, *6, 7*, *9, 10*, *12, 13*, *15*, *17–19*, *21, 22*, *24–26*, *28*, *30*, *32*, *34–36*, *38, 39*, *41*, *43–45*
logical, *8*, *10*, *12*, *31*, *41*, *45*

matrix, *29*

numeric, *3–5*, *7, 8*, *10*, *12–14*, *16–19*, *21–23*, *25, 26*, *28*, *30, 31*, *34–37*, *39*, *41–43*

peaks_find, 14, *16*
peaks_find,ANY,missing-method (peaks_find), 14
peaks_find,numeric,numeric-method (peaks_find), 14
peaks_find-method (peaks_find), 14
peaks_fwhm, *15*, 16
peaks_fwhm,ANY,missing-method (peaks_fwhm), 16
peaks_fwhm,numeric,numeric-method (peaks_fwhm), 16
peaks_fwhm-method (peaks_fwhm), 16

replace_negative, 17, *19*
replace_negative,ANY,missing-method (replace_negative), 17
replace_negative,numeric,numeric-method (replace_negative), 17
replace_negative-method (replace_negative), 17
replace_threshold, *18*, 18
replace_threshold,ANY,missing,function-method (replace_threshold), 18
replace_threshold,ANY,missing,numeric-method (replace_threshold), 18
replace_threshold,numeric,numeric,function-method (replace_threshold), 18
replace_threshold,numeric,numeric,numeric-method (replace_threshold), 18
replace_threshold-method (replace_threshold), 18
resample_bin, 19, *21, 22*
resample_bin,ANY,missing-method (resample_bin), 19
resample_bin,numeric,numeric-method (resample_bin), 19
resample_bin-method (resample_bin), 19
resample_down, *19*, 20, *22*
resample_down,ANY,missing-method (resample_down), 20
resample_down,numeric,numeric-method (resample_down), 20
resample_down-method (resample_down), 20
resample_interpolate, *19*, *21*, 22

resample_interpolate,ANY,missing-method (resample_interpolate), 22
resample_interpolate,numeric,numeric-method (resample_interpolate), 22
resample_interpolate-method (resample_interpolate), 22
rescale_area, 23, *25*, *27, 28*
rescale_area,ANY,missing-method (rescale_area), 23
rescale_area,numeric,numeric-method (rescale_area), 23
rescale_area-method (rescale_area), 23
rescale_max (rescale_range), 24
rescale_max,ANY,missing-method (rescale_range), 24
rescale_max,numeric,numeric-method (rescale_range), 24
rescale_max-method (rescale_range), 24
rescale_min (rescale_range), 24
rescale_min,ANY,missing-method (rescale_range), 24
rescale_min,numeric,numeric-method (rescale_range), 24
rescale_min-method (rescale_range), 24
rescale_range, *24*, 24, *27, 28*
rescale_range,ANY,missing-method (rescale_range), 24
rescale_range,numeric,numeric-method (rescale_range), 24
rescale_range-method (rescale_range), 24
rescale_total, *24, 25*, 26, *28*
rescale_total,ANY,missing-method (rescale_total), 26
rescale_total,numeric,numeric-method (rescale_total), 26
rescale_total-method (rescale_total), 26
rescale_transform, *24, 25*, *27*, 27
rescale_transform,ANY,missing-method (rescale_transform), 27
rescale_transform,numeric,numeric-method (rescale_transform), 27
rescale_transform-method (rescale_transform), 27

signal_bind, 28, *30*, *32–34*, *43*
signal_bind,ANY-method (signal_bind), 28
signal_bind-method (signal_bind), 28
signal_correct, *29*, 30, *32–34*, *43*
signal_correct(), *3*, *5–7*, *9, 10*
signal_correct,ANY,missing-method (signal_correct), 30
signal_correct,numeric,numeric-method (signal_correct), 30
signal_correct-method (signal_correct), 30
signal_drift, *29, 30*, 31, *33, 34*, *43*
signal_drift,ANY,missing,ANY-method (signal_drift), 31
signal_drift,numeric,numeric,numeric-method (signal_drift), 31
signal_drift-method (signal_drift), 31
signal_mean, *29, 30*, *32*, 32, *34*, *43*
signal_mean,ANY-method (signal_mean), 32
signal_mean-method (signal_mean), 32
signal_select (subset), 42
signal_select,ANY,missing-method (subset), 42
signal_select,numeric,numeric-method (subset), 42
signal_select-method (subset), 42
signal_shift, *29, 30*, *32, 33*, 33, *43*
signal_shift,ANY,missing-method (signal_shift), 33
signal_shift,numeric,numeric-method (signal_shift), 33
signal_shift-method (signal_shift), 33
signal_slice (subset), 42
signal_slice,ANY,missing-method (subset), 42
signal_slice,numeric,numeric-method (subset), 42
signal_slice-method (subset), 42
smooth_loess, 34, *36*, *38*, *40, 41*
smooth_loess,ANY,missing-method (smooth_loess), 34
smooth_loess,numeric,numeric-method (smooth_loess), 34
smooth_loess-method (smooth_loess), 34
smooth_rectangular, *35*, 36, *38*, *40, 41*
smooth_rectangular,ANY,missing-method (smooth_rectangular), 36
smooth_rectangular,numeric,numeric-method (smooth_rectangular), 36
smooth_rectangular-method (smooth_rectangular), 36
smooth_savitzky, *35, 36*, 37, *40, 41*
smooth_savitzky,ANY,missing-method

(smooth_savitzky), 37
smooth_savitzky,numeric,numeric-method
(smooth_savitzky), 37
smooth_savitzky-method
(smooth_savitzky), 37
smooth_triangular, *35*, *36*, *38*, 39, *41*
smooth_triangular,ANY,missing-method
(smooth_triangular), 39
smooth_triangular,numeric,numeric-method
(smooth_triangular), 39
smooth_triangular-method
(smooth_triangular), 39
smooth_whittaker, *35*, *36*, *38*, *40*, 40
smooth_whittaker,ANY,missing-method
(smooth_whittaker), 40
smooth_whittaker,numeric,numeric-method
(smooth_whittaker), 40
smooth_whittaker-method
(smooth_whittaker), 40
stats::approx(), *22*
stats::loess(), *35*
stats::smooth.spline(), *8*
subset, *29*, *30*, *32–34*, 42
sum(), *19*

window_sliding, 43, *45*
window_sliding(), *7*, *36*, *39*
window_sliding,integer,integer-method
(window_sliding), 43
window_sliding,numeric,numeric-method
(window_sliding), 43
window_sliding-method (window_sliding), 43
window_tumbling, *44*, 45
window_tumbling(), *4*, *19*
window_tumbling,integer,integer-method
(window_tumbling), 45
window_tumbling,numeric,numeric-method
(window_tumbling), 45
window_tumbling-method
(window_tumbling), 45

XRD, *11*, *14*, 46