Package 'gamma'

May 8, 2026

Title Dose Rate Estimation from in-Situ Gamma-Ray Spectrometry Measurements

Version 1.1.0

Maintainer Archéosciences Bordeaux <services-archeosciences@u-bordeaux-montaigne.fr>

Description Process in-situ Gamma-Ray Spectrometry for Luminescence Dating. This package allows to import, inspect and correct the energy shifts of gamma-ray spectra. It provides methods for estimating the gamma dose rate by the use of a calibration curve as described in Mercier and Falguères (2007). The package only supports Canberra CNF and TKA and Kromek SPE files.

License GPL-3

URL https://crp2a.github.io/gamma/, https://github.com/crp2a/gamma

BugReports https://github.com/crp2a/gamma/issues

Depends R (>= 3.5)

Imports ggplot2, graphics, IsoplotR, methods, rlang, rxylib, stats, tools, utils

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0), vdiffr (>= 1.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData false

RoxygenNote 7.3.2

Collate 'AllClasses.R' 'AllGenerics.R' 'baseline.R' 'baseline_linear.R' 'baseline_rubberband.R' 'baseline_snip.R' 'coerce.R' 'data.R' 'dose_fit.R' 'dose_predict.R' 'energy_calibrate.R' 'gamma-package.R' 'initialize.R' 'mutators.R' 'operators.R' 'peaks_find.R' 'peaks_search.R' 'plot.R' 'read.R' 'show.R' 'signal_integrate.R' 'signal_slice.R' 'signal_split.R' 'signal_stabilize.R' 'smooth.R' 'smooth_rectangular.R' 'smooth_savitzky.R'

'smooth_triangular.R' 'subset.R' 'summarize.R' 'utilities.R'
'validate.R' 'zzz.R'

NeedsCompilation no

Author Nicolas Frerebeau [aut] (ORCID:
<https://orcid.org/0000-0001-5759-4944>),
Brice Lebrun [aut] (ORCID: <https://orcid.org/0000-0001-7503-8685>),
Guilhem Paradol [aut] (ORCID: <https://orcid.org/0000-0002-8561-4903>),
Magali Rizza [ctb] (ORCID: <https://orcid.org/0000-0003-2364-5621>),
Christelle Lahaye [ctb] (ORCID:
<https://orcid.org/0000-0003-2215-9015>),
Sebastian Kreutzer [aut] (ORCID:
<https://orcid.org/0000-0002-0734-2199>),
Archéosciences Bordeaux [cre],
Université Bordeaux Montaigne [cph, fnd],
CNRS [fnd],
LabEx Sciences archéologiques de Bordeaux [fnd],
Idex Aix-Marseille [fnd]

Repository CRAN

Date/Publication 2024-09-23 22:00:12 UTC

Contents

AIX_NaI_1 . . . 3
baseline . . . 3
Baseline-class . . . 6
BDX_LaBr_1 . . . 7
CalibrationCurve-class . . . 8
clermont . . . 9
clermont_2024 . . . 10
coerce . . . 10
doserate . . . 11
energy . . . 15
GammaSpectra-class . . . 18
GammaSpectrum-class . . . 19
mutator . . . 21
operator . . . 25
PeakPosition-class . . . 26
peaks_find . . . 27
peaks_search . . . 29
plot . . . 30
read . . . 32
signal_integrate . . . 33
signal_slice . . . 35
signal_split . . . 37
signal_stabilize . . . 38
smooth . . . 39

subset 41
summarise 42

Index **44**

`AIX_NaI_1` *CEREGE Calibration Curve (NaI)*

Description

CEREGE Calibration Curve (NaI)

Usage

```
data(AIX_NaI_1)
```

Format

An object of class CalibrationCurve.

Laboratory	CEREGE
Instrument	Canberra Inspector 1000
Detector	NaI
Authors	CEREGE Luminescence Team

See Also

Other datasets: `BDX_LaBr_1`, `clermont`, `clermont_2024`

Examples

```
## Load the curve
data(AIX_NaI_1, package = "gamma")
plot(AIX_NaI_1)
```

`baseline` *Baseline Estimation and Removal*

Description

Baseline Estimation and Removal

Usage

```
signal_baseline(object, ...)

signal_correct(object, ...)

baseline_snip(object, ...)

baseline_rubberband(object, ...)

baseline_linear(object, ...)

## S4 method for signature 'GammaSpectrum'
signal_baseline(object, method = c("SNIP", "rubberband", "linear"), ...)

## S4 method for signature 'GammaSpectra'
signal_baseline(object, method = c("SNIP", "rubberband", "linear"), ...)

## S4 method for signature 'GammaSpectrum'
signal_correct(object, method = c("SNIP", "rubberband", "linear"), ...)

## S4 method for signature 'GammaSpectra'
signal_correct(object, method = c("SNIP", "rubberband", "linear"), ...)

## S4 method for signature 'GammaSpectrum'
baseline_linear(object, from = NULL, to = NULL)

## S4 method for signature 'GammaSpectra'
baseline_linear(object, from = NULL, to = NULL)

## S4 method for signature 'GammaSpectrum'
baseline_rubberband(object, noise = 0, spline = TRUE, ...)

## S4 method for signature 'GammaSpectra'
baseline_rubberband(object, noise = 0, spline = TRUE, ...)

## S4 method for signature 'GammaSpectrum'
baseline_snip(object, LLS = FALSE, decreasing = FALSE, n = 100, ...)

## S4 method for signature 'GammaSpectra'
baseline_snip(object, LLS = FALSE, decreasing = FALSE, n = 100, ...)
```

Arguments

object	A GammaSpectrum or GammaSpectra object.
...	Extra parameters to be passed to further methods.
method	A character string specifying the method to be used for baseline estimation (see details). Any unambiguous substring can be given.

from	An integer giving the first channel to be used for linear interpolation. If NULL (the default), channel 1 is used. Only used if method is "linear".
to	An integer giving the last channel to be used for linear interpolation. If NULL (the default), channel max is used. Only used if method is "linear".
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".
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? Only used if method is "SNIP".
n	An integer value giving the number of iterations. Only used if method is "SNIP".

Details

The following methods are available for baseline estimation:

SNIP Sensitive Nonlinear Iterative Peak clipping algorithm.

rubberband 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.

linear Linear baseline estimation.

Value

- baseline_*() returns a BaseLine object.
- signal_correct() returns a corrected GammaSpectrum or GammaSpectra object (same as object).

Note

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

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.

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

Other signal processing: `peaks_find()`, `peaks_search()`, `signal_integrate()`, `signal_slice()`, `signal_split()`, `signal_stabilize()`, `smooth()`

Examples

```
## Import a CNF file
spc_file <- system.file("extdata/LaBr.CNF", package = "gamma")
spc <- read(spc_file)

## Remove the first 35 channels
spc <- signal_slice(spc, -c(1:35))

## Linear baseline
bsl_linear <- baseline_linear(spc, from = 250, to = 750)
plot(spc, bsl_linear)

## SNIP baseline
bsl_snip <- baseline_snip(spc, LLS = FALSE, decreasing = FALSE, n = 100)
plot(spc, bsl_snip)

## Rubberband baseline
bsl_rubber <- baseline_rubberband(spc)
plot(spc, bsl_rubber)

## Remove baseline
spc_clean1 <- signal_correct(spc)
spc_clean2 <- spc - bsl_snip
all(spc_clean1 == spc_clean2)

plot(spc_clean1)
```

Baseline-class *An S4 Class to Represent a Spectrum Baseline*

Description

An S4 Class to Represent a Spectrum Baseline

Note

This class extends the GammaSpectrum class.

Author(s)

N. Frerebeau

See Also

Other class: CalibrationCurve-class, GammaSpectra-class, GammaSpectrum-class, PeakPosition-class, coerce()

Examples

```
## Import a CNF file
spc_file <- system.file("extdata/LaBr.CNF", package = "gamma")
spc <- read(spc_file)

## Remove the first 35 channels
spc <- signal_slice(spc, -c(1:35))

## Linear baseline
bsl_linear <- baseline_linear(spc, from = 250, to = 750)
plot(spc, bsl_linear)

## SNIP baseline
bsl_snip <- baseline_snip(spc, LLS = FALSE, decreasing = FALSE, n = 100)
plot(spc, bsl_snip)

## Rubberband baseline
bsl_rubber <- baseline_rubberband(spc)
plot(spc, bsl_rubber)

## Remove baseline
spc_clean1 <- signal_correct(spc)
spc_clean2 <- spc - bsl_snip
all(spc_clean1 == spc_clean2)

plot(spc_clean1)
```

BDX_LaBr_1 *CRP2A Calibration Curve (LaBr)*

Description

CRP2A Calibration Curve (LaBr)

Usage

```
data(BDX_LaBr_1)
```

Format

An object of class CalibrationCurve.

Laboratory	IRAMAT-CRP2A (UMR 5060)
Instrument	Canberra Inspector 1000
Detector	LaBr
Authors	CRP2A Luminescence Team

See Also

Other datasets: `AIX_NaI_1`, `clermont`, `clermont_2024`

Examples

```
## Load the curve
data(BDX_LaBr_1, package = "gamma")
plot(BDX_LaBr_1)
```

CalibrationCurve-class

An S4 class to Represent a Dose Rate Calibration Curve

Description

An S4 class to Represent a Dose Rate Calibration Curve

Slots

`Ni` A DoseRateModel object.

`NiEi` A DoseRateModel object.

`details` A `list` of length-one vector giving the curve metadata.

`slope` A `numeric` vector.

`intercept` A `numeric` vector.

`covariance` A `numeric` vector.

`MSWD` A `numeric` vector.

`df` A `numeric` vector.

`p_value` A `numeric` vector.

`data` A `data.frame`.

`range` A `numeric` vector.

`background` A `numeric` vector.

Subset

In the code snippets below, `x` is a `CalibrationCurve` object.

`x[[i]]` Extracts information from a slot selected by subscript `i`. `i` is a `character` vector of length one.

Author(s)

N. Frerebeau

See Also

Other class: `Baseline-class`, `GammaSpectra-class`, `GammaSpectrum-class`, `PeakPosition-class`, `coerce()`

---

clermont *Clermont Reference Data*

Description

Clermont Reference Data

Usage

```
data("clermont")
```

Format

A `data.frame` with tabulated data and conversion factor reference.

Source

Guérin, G., Mercier, N. & Adamiec, G. (2011). Dose-Rate Conversion Factors: Update. *Ancient TL*, 29(1), p. 5-8.

Miallier, D., Guérin, G., Mercier, N., Pilleyre, T. & Sanzelle, S. (2009). The Clermont Radiometric Reference Rocks: A Convenient Tool for Dosimetric Purposes. *Ancient TL*, 27(2), p. 37-44.

See Also

Other datasets: `AIX_NaI_1`, `BDX_LaBr_1`, `clermont_2024`

clermont_2024 *Clermont Reference Data 2024*

Description

An updated version for the `clermont` dataset with additional columns for the applied dose rate conversion factors and their reference. The dataset also contains gamma dose rate values and uncertainties for different published conversion factors. Please note that the values for the conversion factors applied in the original `clermont` dataset differ slightly due to rounding errors.

Usage

```
data("clermont_2024")
```

Format

A `data.frame` with tabulated data and conversion factor reference.

Source

Adamiec, G. & Aitken, M.J. (1998). Dose-rate conversion factors: update. *Ancient TL*, 16, p. 37–50.

Cresswell, A.J., Carter, J. & Sanderson, D.C.W. (2018). Dose rate conversion parameters: Assessment of nuclear data. *Radiation Measurements*, 120, p. 195–201.

Guérin, G., Mercier, N. & Adamiec, G. (2011). Dose-Rate Conversion Factors: Update. *Ancient TL*, 29(1), p. 5-8.

Liritzis, I., Stamoulis, K., Papachristodoulou, C. & Ioannides, K. (2013). A re-evaluation of radiation dose-rate conversion factors. *Mediterranean Archaeology and Archaeometry*, 12, p. 1–15.

Miallier, D., Guérin, G., Mercier, N., Pilleyre, T. & Sanzelle, S. (2009). The Clermont Radiometric Reference Rocks: A Convenient Tool for Dosimetric Purposes. *Ancient TL*, 27(2), p. 37-44.

See Also

Other datasets: `AIX_NaI_1`, `BDX_LaBr_1`, `clermont`

coerce *Coerce*

Description

Coerce

Usage

```
## S3 method for class 'GammaSpectrum'
as.matrix(x, ...)
```

Arguments

x	An object to be coerced.
...	Currently not used.

Value

A coerced object.

Author(s)

N. Frerebeau

See Also

Other class: Baseline-class, CalibrationCurve-class, GammaSpectra-class, GammaSpectrum-class, PeakPosition-class

Examples

```
## Import a Canberra CNF file
spc_file <- system.file("extdata/LaBr.CNF", package = "gamma")
spc <- read(spc_file)

## Coerce
mtx <- as.matrix(spc)
df <- as.data.frame(spc)
head(df)
```

---

doserate *Dose Rate Estimation*

Description

- dose_fit() builds a calibration curve for gamma dose rate estimation.
- dose_predict() predicts in situ gamma dose rate.

Usage

```
dose_fit(object, background, doses, ...)

dose_predict(object, spectrum, ...)

## S4 method for signature 'GammaSpectra,GammaSpectrumOrNumeric,matrix'
dose_fit(
  object,
  background,
  doses,
```

```
  range_Ni,
  range_NiEi,
  details = list(authors = "", date = Sys.time())
)

## S4 method for signature 'GammaSpectra,GammaSpectrumOrNumeric,data.frame'
dose_fit(
  object,
  background,
  doses,
  range_Ni,
  range_NiEi,
  details = list(authors = "", date = Sys.time())
)

## S4 method for signature 'CalibrationCurve,missing'
dose_predict(
  object,
  sigma = 1,
  epsilon = 0.015,
  water_content = NULL,
  use_MC = FALSE
)

## S4 method for signature 'CalibrationCurve,GammaSpectrum'
dose_predict(
  object,
  spectrum,
  sigma = 1,
  epsilon = 0.015,
  water_content = NULL,
  use_MC = FALSE
)

## S4 method for signature 'CalibrationCurve,GammaSpectra'
dose_predict(
  object,
  spectrum,
  sigma = 1,
  epsilon = 0.015,
  water_content = NULL,
  use_MC = FALSE
)
```

Arguments

object	A GammaSpectra or CalibrationCurve object.
background	A GammaSpectrum object or a length-two numeric vector giving the back-

ground noise integration value and error, respectively. If no background subtraction is wanted, you can set `background = c(0,0,)`

`doses` A `matrix` or `data.frame` object with gamma dose values and uncertainties. The row names must match the names of the spectrum.

`...` Currently not used.

`spectrum` An optional GammaSpectrum or GammaSpectra object in which to look for variables with which to predict. If omitted, the fitted values are used.

`range_Ni`, `range_NiEi`
A length-two `numeric` vector giving the energy range to integrate within (in keV).

`details` A `list` of length-one vector specifying additional informations about the instrument for which the curve is built.

`sigma` A `numeric` value giving the confidence level of which the error from the slope is considered in the final uncertainty calculation

`epsilon` A `numeric` value giving an extra relative error term, introduced by the calibration of the energy scale of the spectrum, e.g., `0.015` for an additional 1.5% error

`water_content` `numeric` or `matrix` gravimetric field water content to correct the gamma-dose rate to using the correction factor by Aitken (1985) to obtain the dry gamma-dose rate. Example: `c(0.05,0.0001)` for water content of 5% +/- 0.01 %. The default is `NULL` (nothing is corrected). The correction only works on the final dose rate. For more information see details.

`use_MC` A `logical` parameter, enabling/disabling Monte Carlo simulations for estimating the dose rate uncertainty

Details

To estimate the gamma dose rate, one of the calibration curves distributed with this package can be used. These built-in curves are in use in several luminescence dating laboratories and can be used to replicate published results. As these curves are instrument specific, the user may have to build its own curve.

The construction of a calibration curve requires a set of reference spectra for which the gamma dose rate is known and a background noise measurement. First, each reference spectrum is integrated over a given interval, then normalized to active time and corrected for background noise. The dose rate is finally modelled by the integrated signal value used as a linear predictor (York *et al.*, 2004).

Value

- `dose_fit()` returns a CalibrationCurve object.
- `dose_predict()` returns a `data.frame` with the following columns:

  `name` (`character`) the name of the spectra.

  `signal_Ni` (`numeric`) the integrated signal value (according to the value of `threshold`; see `signal_integrate()`) for `energy = FALSE`

  `signal_err_Ni` (`numeric`) the integrated signal error value (according to the value of `threshold`; see `signal_integrate()`) for `energy = FALSE`.

  `dose_Ni` (`numeric`) the predicted gamma dose rate for `energy = FALSE`.

`dose_err_Ni` (numeric) the predicted gamma dose rate error for `energy = FALSE`.

`signal_Ni` (numeric) the integrated signal value (according to the value of `threshold`; see `signal_integrate()`).

`signal_err_NiEi` (numeric) the integrated signal error value (according to the value of `threshold`; see `signal_integrate()`) for `energy = TRUE`.

`dose_NiEi` (numeric) the predicted gamma dose rate for `energy = TRUE`.

`dose_err_NiEi` (numeric) the predicted gamma dose rate error for `energy = TRUE`.

`dose_final` (numeric) the predicted final gamma dose rate as the mean of `dose_Ni` and `dose_NiEi`

`dose_err_final` (numeric) the predicted final gamma dose rate error as $SE(\dot{D}_\gamma) = \sqrt{(\frac{SE(\dot{D}_{\gamma\mathrm{Ni}})}{\dot{D}_{\gamma\mathrm{Ni}}})^2 + (\frac{SE(\dot{D}_{\gamma\mathrm{NiEi}})}{\dot{D}_{\gamma\mathrm{NiEi}}})^2}$

Uncertainty calculation of the gamma-dose rate

The analytical uncertainties of the final gamma-dose rate ($SE(\dot{D}_\gamma)$) are calculated as follows:

$$\sigma_{\dot{D}_\gamma} = \sqrt{((\frac{m_\delta s}{m})^2 + (\frac{s_\delta}{s})^2 + \epsilon^2)}$$

with m and m_δ being the slope of the fit an its uncertainty, σ the error scaler for the slope uncertainty, s and s_δ the integrated signal and its uncertainty, and ϵ an additional relative uncertainty term that can be set by the user using the argument `epsilon`.

If the parameter `use_MC` is set to `TRUE`, the a Monte Carlo sampling approach is chosen to approximate the uncertainties on the dose rate:

$$\sigma_{\dot{D}_\gamma} := \sqrt{((\frac{SD(\mathcal{N}(\mu_{slope}, \sigma_{slope}) \times \mathcal{N}(\mu_{signal}, \sigma_{signal}) + \mathcal{N}(\mu_{intercept}, \sigma_{intercept})) * \rho}{\dot{D}_\gamma})^2 + \epsilon^2) * \dot{D}_\gamma}$$

rho is the parameter `sigma` provided with the function call, SD equals the the call to `sd()`, i.e. the calculation of the standard deviation. To achieve a good Gaussian normal approximation with sample 1+e06 times (the values is fixed).

Water content correction

If gamma-dose rates are measured in the field, they are measured at "as-is" conditions. In dating studies, however, using the dry dose rate is often more desirable to model the long-term effect of different assumptions for the water content. If the parameter `water_content`, either as numeric vector or as matrix with the number of rows equal to the number of processed spectra, if different values are desired, the **final** gamma-dose rate is corrected for the water content provided. Final uncertainties are obtained using the square root of the summed squared relative uncertainties of the dose rate and the water content.

A word of caution: When estimating the water content in the laboratory, the water analytical uncertainty is usually minimal, and it does not make sense to correct with a relative water content of, e.g., c(0.02,0.02) (2% +/- 2%) because this massively inflates the final dose rate error.

Note

See `vignette(doserate)` for a reproducible example.

Author(s)

N. Frerebeau

References

Aitken, M.J. (1985). *Thermoluminescence dating*. London: Academic Press.

Mercier, N. & Falguères, C. (2007). Field Gamma Dose-Rate Measurement with a NaI(Tl) Detector: Re-Evaluation of the "Threshold" Technique. *Ancient TL*, 25(1), p. 1-4.

York, D., Evensen, N. M., Martínez, M. L. & De Basabe Delgado, J. (2004). Unified Equations for the Slope, Intercept, and Standard Errors of the Best Straight Line. *American Journal of Physics*, 72(3), p. 367-75. doi:10.1119/1.1632486.

See Also

`signal_integrate()`

Examples

```
## Import CNF files
## Spectra
spc_dir <- system.file("extdata/BDX_LaBr_1/calibration", package = "gamma")
spc <- read(spc_dir)

## Background
bkg_dir <- system.file("extdata/BDX_LaBr_1/background", package = "gamma")
bkg <- read(bkg_dir)

## Get dose rate values
data("clermont")
(doses <- clermont[, c("gamma_dose", "gamma_error")])

## Build the calibration curve
calib_curve <- dose_fit(spc, bkg, doses,
                        range_Ni = c(300, 2800),
                        range_NiEi = c(165, 2800))

## Plot the curve
plot(calib_curve, threshold = "Ni")

## Estimate gamma dose rates
dose_predict(calib_curve, spc)
```

---

energy *Energy Scale Calibration*

Description

Calibrates the energy scale of a gamma spectrum.

Usage

```
energy_calibrate(object, lines, ...)

has_energy(object)

has_calibration(object)

## S4 method for signature 'GammaSpectrum,lm'
energy_calibrate(object, lines, ...)

## S4 method for signature 'GammaSpectrum,GammaSpectrum'
energy_calibrate(object, lines, ...)

## S4 method for signature 'GammaSpectrum,CalibrationCurve'
energy_calibrate(object, lines, ...)

## S4 method for signature 'GammaSpectrum,list'
energy_calibrate(object, lines, ...)

## S4 method for signature 'GammaSpectrum,PeakPosition'
energy_calibrate(object, lines, ...)

## S4 method for signature 'GammaSpectra,list'
energy_calibrate(object, lines, ...)

## S4 method for signature 'GammaSpectra,PeakPosition'
energy_calibrate(object, lines, ...)

## S4 method for signature 'GammaSpectra,lm'
energy_calibrate(object, lines, ...)

## S4 method for signature 'GammaSpectra,GammaSpectrum'
energy_calibrate(object, lines, ...)

## S4 method for signature 'GammaSpectra,CalibrationCurve'
energy_calibrate(object, lines, ...)

## S4 method for signature 'GammaSpectrum'
has_energy(object)

## S4 method for signature 'GammaSpectra'
has_energy(object)

## S4 method for signature 'GammaSpectrum'
has_calibration(object)

## S4 method for signature 'GammaSpectra'
has_calibration(object)
```

```
## S4 method for signature 'CalibrationCurve'
has_calibration(object)
```

Arguments

object	A GammaSpectrum or GammaSpectra object.
lines	A PeakPosition object or a list of length two. If a list is provided, each element must be a named numeric vector giving the observed peak position ("channel") and the corresponding expected "energy" value (in keV). Alternatively, the function accepts the stats::lm object from a another calibration or a GammaSpectrum object from which such calibration is copied. With this, energy calibrations can be transferred from one object to another.
...	Currently not used.

Details

The energy calibration of a spectrum is the most tricky part. To do this, the user must specify the position of at least three observed peaks and the corresponding energy value (in keV). A second order polynomial model is fitted on these energy *vs* channel values, then used to predict the new energy scale of the spectrum.

The package allows to provide the channel-energy pairs to be use. However, the spectrum can be noisy so it is difficult to properly determine the peak channel. In this case, a better approach may be to pre-process the spectrum (variance-stabilization, smoothing and baseline correction) and perform a peak detection. Once the identified peaks are satisfactory, you can set the corresponding energy values (in keV) and use these lines to calibrate the energy scale of the spectrum.

Regardless of the approach you choose, it is strongly recommended to check the result before proceeding.

Value

- energy_calibrate() returns either a GammaSpectrum or a GammaSpectra object depending on the input#
- has_energy() and has_calibration() return a logical vector.

Author(s)

N. Frerebeau

Examples

```
## Import a CNF file
spc_file <- system.file("extdata/LaBr.TKA", package = "gamma")
(spc <- read(spc_file))

## Set peak positions (channel) and expected energy values
calib_lines <- list(
  channel = c(86, 495, 879),
  energy = c(238, 1461, 2615)
```

```
)

## Adjust the energy scale
(spc1 <- energy_calibrate(spc, lines = calib_lines))

## Inspect results
plot(spc1, xaxis = "energy", yaxis = "count") +
  ggplot2::geom_vline(xintercept = c(238, 1461, 2615), linetype = 3)
```

`GammaSpectra-class` *An S4 Class to Represent a Collection of Gamma Sectra*

Description

Represents a collection of spectra of gamma ray spectrometry measurements.

Details

This class extends the base `list` and can only contains GammaSpectrum objects.

Access

In the code snippets below, `x` is a `GammaSpectra` object.

`length(x)` Get the number of elements in `x`.

`lengths(x)` Get the number of channels in each element of `x`.

`get_names(x)`, `set_names(x) <- value` Retrieves or sets the names of `x` according to `value`.

`get_hash(x)` Get the MD5 hash of the raw data files.

`get_channels(x)` Get the number of channels in each element of `x`.

`get_counts(x)` Get the counts of each element of `x`.

`get_energy(x)` Get the energy range of each element of `x`.

`get_rates(x)` Get the count rates of each element of `x`.

Subset

In the code snippets below, `x` is a `GammaSpectra` object.

`x[i]` Extracts the elements selected by subscript `i`. `i` can be `missing` or `NULL`, `numeric` or `character` vector or a `factor`. Returns a new `GammaSpectra` object.

`x[i, j]` Like the above but allows to select a slot thru `j` (see examples). `j` is a `character` vector of length one. Returns a `list`.

`x[[i]]` Extracts the elements selected by subscript `i`. `i` can be a `numeric` or `character` vector of length one. Returns the corresponding GammaSpectrum object.

Author(s)

N. Frerebeau

See Also

Other class: Baseline-class, CalibrationCurve-class, GammaSpectrum-class, PeakPosition-class, coerce()

Examples

```
## Import all CNF files in a given directory
spc_dir <- system.file("extdata/BDX_LaBr_1/calibration", package = "gamma")
(spc <- read(spc_dir))

## Access
get_hash(spc)
get_names(spc)
get_livetime(spc)
get_realtime(spc)

lengths(spc)
range_energy(spc)

## Subset
spc[] # All spectra
spc[NULL] # All spectra
spc[1] # The first spectrum
spc[-6] # Delete the sixth spectrum
spc[1:3] # The first three spectra
spc[c(1, 3)] # The first and third spectra
spc["BRIQUE"] # The spectrum named 'BRIQUE'
spc[c("BRIQUE", "C347")] # The spectra named 'BRIQUE' and 'C347'
spc[1:3, "energy"] # The slot 'energy' of the first three spectra
spc[[1]]
spc[["BRIQUE"]]
```

GammaSpectrum-class *An S4 Class to Represent a Gamma Spectrum*

Description

Represents a single spectrum of a gamma ray spectrometry measurement.

Slots

`hash` A character string giving the 32-byte MD5 hash of the imported file.

`name` A character string the measurement reference.

`date` A POSIXct element giving the measurement date and time.

`instrument` A character string giving the instrument name.

`file_format` A character string giving the format of the imported file.

`live_time` A numeric value.

`real_time` A `numeric` value.

`channel` A `integer` vector giving the channel number. Numeric values are coerced to integer as by `as.integer()` (and hence truncated towards zero).

`energy` A `numeric` vector giving the gamma ray's energy (in keV).

`count` A `numeric` vector giving the counts number for each channel. Numeric values are coerced to integer as by `as.integer()` (and hence truncated towards zero).

`rate` A `numeric` vector the count rate ($1/s$) for each channel.

`calibration` A linear model used for energy scale calibration (see `energy_calibrate()`).

Access

In the code snippets below, `x` is a `GammaSpectrum` object.

`length(x)` Get number of channel in `x`.

`get_hash(x)` Get the MD5 hash of the raw data file.

`get_names(x)`, `set_names(x) <- value` Retrieves or sets the name of `x` according to `value`.

`get_channels(x)` Get the number of channels in `x`.

`get_counts(x)` Get the counts of `x`.

`get_energy(x)` Get the energy range of `x`.

`get_rates(x)` Get the count rates of `x`.

Coerce

In the code snippets below, `x` is a `GammaSpectrum` object.

`as.matrix(x)` Coerces `x` to a `matrix`.

`as.data.frame(x)` Coerces `x` to a `data.frame`.

Subset

In the code snippets below, `x` is a `GammaSpectrum` object.

`x[[i]]` Extracts information from a slot selected by subscript `i`. `i` is a `character` vector of length one and will be matched to the name of the slots.

Note

This class retains copy construction.

Author(s)

N. Frerebeau

See Also

Other class: `Baseline-class`, `CalibrationCurve-class`, `GammaSpectra-class`, `PeakPosition-class`, `coerce()`

Examples

```
## Import a Canberra CNF file
spc_file <- system.file("extdata/LaBr.CNF", package = "gamma")
(spc <- read(spc_file))

## Access
get_hash(spc)
get_names(spc)
get_livetime(spc)
get_realtime(spc)

length(spc)
range_energy(spc)

## Subset
spc[["date"]]
spc[["instrument"]]
spc[["file_format"]]
```

`mutator` *Get or Set Parts of an Object*

Description

Getters and setters to extract or replace parts of an object.

Usage

```
get_hash(x)

get_names(x)

set_names(x) <- value

get_energy_calibration(x)

set_energy_calibration(x) <- value

get_livetime(x)

get_realtime(x)

get_channels(x)

get_counts(x)

get_rates(x)
```

```
get_energy(x, ...)

set_energy(x) <- value

get_method(x)

set_method(x) <- value

get_residuals(x)

range_channels(x, ...)

range_energy(x, ...)

## S4 method for signature 'GammaSpectrum'
length(x)

## S4 method for signature 'GammaSpectrum'
get_hash(x)

## S4 method for signature 'GammaSpectrum'
get_names(x)

## S4 method for signature 'GammaSpectrum'
get_livetime(x)

## S4 method for signature 'GammaSpectrum'
get_realtime(x)

## S4 method for signature 'GammaSpectrum'
get_channels(x)

## S4 method for signature 'GammaSpectrum'
get_counts(x)

## S4 method for signature 'GammaSpectrum'
get_rates(x)

## S4 method for signature 'GammaSpectrum'
get_energy(x)

## S4 method for signature 'GammaSpectrum'
range_energy(x, na.rm = FALSE)

## S4 method for signature 'GammaSpectrum'
range_channels(x, na.rm = FALSE)

## S4 method for signature 'Baseline'
```

```
get_method(x)

## S4 method for signature 'GammaSpectra'
get_hash(x)

## S4 method for signature 'GammaSpectra'
get_names(x)

## S4 method for signature 'GammaSpectra'
get_livetime(x)

## S4 method for signature 'GammaSpectra'
get_realtime(x)

## S4 method for signature 'GammaSpectra'
get_channels(x)

## S4 method for signature 'GammaSpectra'
get_counts(x)

## S4 method for signature 'GammaSpectra'
get_rates(x)

## S4 method for signature 'GammaSpectra'
get_energy(x)

## S4 method for signature 'GammaSpectra'
range_energy(x, na.rm = FALSE)

## S4 method for signature 'GammaSpectra'
range_channels(x, na.rm = FALSE)

## S4 method for signature 'DoseRateModel'
get_residuals(x)

## S4 method for signature 'PeakPosition'
get_hash(x)

## S4 method for signature 'PeakPosition'
get_channels(x)

## S4 method for signature 'PeakPosition'
get_energy(x, expected = FALSE)

## S4 replacement method for signature 'GammaSpectrum'
set_names(x) <- value

## S4 replacement method for signature 'GammaSpectrum'
```

```
set_energy_calibration(x) <- value

## S4 method for signature 'GammaSpectrum'
get_energy_calibration(x)

## S4 replacement method for signature 'Baseline'
set_method(x) <- value

## S4 replacement method for signature 'GammaSpectra'
set_names(x) <- value

## S4 replacement method for signature 'GammaSpectra'
set_energy_calibration(x) <- value

## S4 method for signature 'GammaSpectra'
get_energy_calibration(x)

## S4 replacement method for signature 'PeakPosition,numeric'
set_energy(x) <- value
```

Arguments

x	An object from which to get or set element(s).
value	A possible value for the element(s) of x.
...	Currently not used.
na.rm	A logical scalar: should NA be omitted?
expected	A logical scalar: should the expected values be returned instead of observed values?

Value

An object of the same sort as x with the new values assigned.

Author(s)

N. Frerebeau

See Also

Other mutator: subset()

operator *Common Operations on GammaSpectrum Objects*

Description

Performs common operations on GammaSpectrum objects.

Usage

```
## S4 method for signature 'GammaSpectrum,GammaSpectrum'
Arith(e1, e2)

## S4 method for signature 'GammaSpectrum,numeric'
Arith(e1, e2)

## S4 method for signature 'GammaSpectrum,GammaSpectrum'
Compare(e1, e2)

## S4 method for signature 'GammaSpectrum,numeric'
Compare(e1, e2)

## S4 method for signature 'GammaSpectrum,GammaSpectrum'
Logic(e1, e2)

## S4 method for signature 'GammaSpectrum,numeric'
Logic(e1, e2)

## S4 method for signature 'GammaSpectrum,logical'
Logic(e1, e2)

## S4 method for signature 'GammaSpectrum'
Math(x)

## S4 method for signature 'GammaSpectrum'
Math2(x, digits)

## S4 method for signature 'GammaSpectrum'
Summary(x, ..., na.rm = FALSE)
```

Arguments

x, e1, e2	An object (typically a GammaSpectrum object).
digits	A length-one numeric vector giving the number of digits to be used in round() or signif().
...	Further arguments passed to or from methods.
na.rm	A logical scalar: should missing values (including NaN) be omitted from the calculations?

Group Generics

GammaSpectrum objects have support for S4 group generic functionality to operate within elements across objects:

`Arith` "+", "-", "*", "^", "\%\%", "\%/\%", "/"

`Compare` "==", ">", "<", "!=", "<=", ">="

`Logic` "&", "|"

`Math` "abs", "sign", "sqrt", "ceiling", "floor", "trunc", "cummax", "cummin", "cumprod", "cumsum", "log", "log10", "log2", "log1p", "acos", "acosh", "asin", "asinh", "atan", "atanh", "exp", "expm1", "cos", "cosh", "cospi", "sin", "sinh", "sinpi", "tan", "tanh", "tanpi", "gamma", "lgamma", "digamma", "trigamma"

`Math2` "round", "signif"

`Ops` "Arith", "Compare", "Logic"

`Summary` "min", "max", "range", "prod", "sum", "any", "all"

Author(s)

N. Frerebeau

Examples

```
## No examples
```

PeakPosition-class *An S4 Class to Represent a Set of Peaks*

Description

An S4 Class to Represent a Set of Peaks

Slots

`hash` A `character` string giving the 32-byte MD5 hash of the imported spectrum file.

`noise_method` A `character` string specifying the method used for peak detection.

`noise_threshold` A length one `numeric` vector giving the noise threshold.

`window` A length one `numeric` vector giving the half-window size.

`channel` A `integer` vector giving the channel number. Numeric values are coerced to integer as by `as.integer()` (and hence truncated towards zero).

`energy_observed` A `numeric` vector giving the observed gamma ray energy (in keV).

`energy_expected` A `numeric` vector giving the expected gamma ray energy (in keV).

Access

In the code snippets below, x is a PeakPosition object.

`get_hash(x)` Get the MD5 hash of the raw data file.

`get_channels(x)` Get the channels of x.

`get_energy(x)`, `set_energy(x) <- value` Retrieves or sets the energy scale of x according to value.

Coerce

In the code snippets below, x is a PeakPosition object.

`as.matrix(x)` Coerces x to a matrix.

`as.data.frame(x)` Coerces x to a data.frame.

Subset

In the code snippets below, x is a PeakPosition object.

`x[[i]]` Extracts information from a slot selected by subscript i. i is a character vector of length one and will be matched to the name of the slots.

Note

This class retains copy construction.

Author(s)

N. Frerebeau

See Also

Other class: Baseline-class, CalibrationCurve-class, GammaSpectra-class, GammaSpectrum-class, coerce()

---

peaks_find *Find Peaks*

Description

Finds local maxima in sequential data.

Usage

```
peaks_find(object, ...)

## S4 method for signature 'GammaSpectrum'
peaks_find(object, method = c("MAD"), SNR = 2, span = NULL, ...)
```

Arguments

object	A GammaSpectrum object.
...	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).
span	An integer giving the half window size (in number of channels). If NULL, 5\ 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.

Value

A PeakPosition object.

Author(s)

N. Frerebeau

See Also

Other signal processing: baseline, peaks_search(), signal_integrate(), signal_slice(), signal_split(), signal_stabilize(), smooth()

Examples

```
## Import a LaBr spectrum
LaBr_file <- system.file("extdata/LaBr.TKA", package = "gamma")
LaBr_spc <- read(LaBr_file)

## Find peaks by channel
(LaBr_pks <- peaks_find(LaBr_spc)) # Ugly
plot(LaBr_spc, LaBr_pks)

## Search peaks by channel
(LaBr_pks <- peaks_search(LaBr_spc, index = c(86L, 207L, 496L), span = 7))
plot(LaBr_spc, LaBr_pks, split = TRUE)

## Import a BEGe spectrum
BEGe_file <- system.file("extdata/BEGe.CNF", package = "gamma")
BEGe_spc <- read(BEGe_file)

## Search peaks by energy
```

```
(BEGe_pks <- peaks_search(BEGe_spc, index = c(47, 63, 911, 1460)))
plot(BEGe_spc, BEGe_pks, split = TRUE)
```

---

peaks_search *Search Peaks*

Description

Search the maxima in sequential data around a given value.

Usage

```
peaks_search(object, index, ...)

## S4 method for signature 'GammaSpectrum,integer'
peaks_search(object, index, span = 10, tolerance = 0.025)

## S4 method for signature 'GammaSpectrum,numeric'
peaks_search(object, index, span = 10, tolerance = 0.025)
```

Arguments

object	A GammaSpectrum object.
index	A vector giving the expected peak position. If index is a numeric vector, peaks are searched by energy (index is assumed to be expressed in keV). If index is an integer vector, peaks are searched by channel.
...	Currently not used.
span	A numeric value giving the half window size for searching. If index is a numeric vector, span is expressed in keV. If index is an integer vector, span is expressed in channel.
tolerance	A numeric value giving the threshold above which a warning/error is raised.

Value

A PeakPosition object.

Author(s)

N. Frerebeau

See Also

Other signal processing: baseline, peaks_find(), signal_integrate(), signal_slice(), signal_split(), signal_stabilize(), smooth()

Examples

```
## Import a LaBr spectrum
LaBr_file <- system.file("extdata/LaBr.TKA", package = "gamma")
LaBr_spc <- read(LaBr_file)

## Find peaks by channel
(LaBr_pks <- peaks_find(LaBr_spc)) # Ugly
plot(LaBr_spc, LaBr_pks)

## Search peaks by channel
(LaBr_pks <- peaks_search(LaBr_spc, index = c(86L, 207L, 496L), span = 7))
plot(LaBr_spc, LaBr_pks, split = TRUE)

## Import a BEGe spectrum
BEGe_file <- system.file("extdata/BEGe.CNF", package = "gamma")
BEGe_spc <- read(BEGe_file)

## Search peaks by energy
(BEGe_pks <- peaks_search(BEGe_spc, index = c(47, 63, 911, 1460)))
plot(BEGe_spc, BEGe_pks, split = TRUE)
```

---

plot *Plot*

Description

Plot

Usage

```
## S4 method for signature 'GammaSpectrum,missing'
plot(x, xaxis = c("channel", "energy"), yaxis = c("count", "rate"), ...)

## S4 method for signature 'GammaSpectrum,Baseline'
plot(x, y, xaxis = c("channel", "energy"), yaxis = c("count", "rate"), ...)

## S4 method for signature 'GammaSpectra,missing'
plot(
  x,
  xaxis = c("channel", "energy"),
  yaxis = c("count", "rate"),
  select = NULL,
  facet = FALSE,
  nrow = c("fixed", "auto")
)

## S4 method for signature 'GammaSpectrum,PeakPosition'
plot(x, y, split = FALSE, span = 25)
```

```
## S4 method for signature 'CalibrationCurve,missing'
plot(
  x,
  error_ellipse = TRUE,
  error_bar = FALSE,
  energy = FALSE,
  level = 0.95,
  n = 50,
  ...
)
```

Arguments

x, y	Objects to be plotted.
xaxis, yaxis	A character string specifying the data to be plotted along each axis. It must be one of "energy" or "channel" (x axis) and "counts" or "rate" (y axis). Any unambiguous substring can be given.
...	Currently not used.
select	A numeric or character vector giving the selection of the spectrum that are drawn.
facet	A logical scalar: should a matrix of panels defined by spectrum be drawn?
nrow	A character string specifying the number of rows. It must be one of "fixed" or "auto". Any unambiguous substring can be given. Only used if facet is TRUE.
split	A logical scalar: should.
span	An integer giving the half window size (in number of channels). Only used if split is TRUE.
error_ellipse	A logical scalar: should error ellipses be plotted?
error_bar	A logical scalar: should error bars be plotted?
energy	A logical scalar plotting the count threshold value or the energy threshold value
level	length-one numeric vector giving the the probability cutoff for the error ellipses.
n	A length-one numeric vector giving the resolution of the error ellipses.

Value

A ggplot2::ggplot object.

Author(s)

N. Frerebeau

See Also

IsoplotR::ellipse(), IsoplotR::isochron()

Examples

```
# Import CNF files
spc_dir <- system.file("extdata/BDX_LaBr_1/calibration", package = "gamma")
spc <- read(spc_dir)

# Plot all spectra
plot(spc, yaxis = "rate", facet = FALSE) +
  ggplot2::theme_bw()

# Plot the spectrum named 'BRIQUE'
plot(spc, xaxis = "energy", yaxis = "count", select = "BRIQUE") +
  ggplot2::theme_bw()

# Plot the first three spectra
plot(spc, xaxis = "channel", yaxis = "rate", select = 1:3, facet = TRUE) +
  ggplot2::theme_bw()
```

---

read *Data Input*

Description

Reads a gamma ray spectrum file.

Usage

```
read(file, ...)

## S4 method for signature 'character'
read(file, extensions = c("cnf", "tka", "spe"), ...)
```

Arguments

file	A character string giving the path of files to be imported.
...	Extra parameters to be passed to rxylib::read_xyData().
extensions	A character vector specifying the possible file extensions. It must be one or more of "cnf", "tka", "spe".

Value

A GammaSpectra object if more than one spectrum are imported at once, else a GammaSpectrum object.

Note

Supports Canberra CNF and TKA and Kromek SPE files.

Author(s)

N. Frerebeau

See Also

rxylib::read_xyData()

Other IO: summarise()

Examples

```
## Import a Canberra CNF file
cnf_file <- system.file("extdata/LaBr.CNF", package = "gamma")
(cnf_spc <- read(cnf_file))

## Import a TKA file
tka_file <- system.file("extdata/LaBr.TKA", package = "gamma")
(tka_spc <- read(tka_file))

## Import all files in a given directory
spc_dir <- system.file("extdata/BDX_LaBr_1/calibration", package = "gamma")
(spc <- read(spc_dir))
```

---

signal_integrate *Signal Integration*

Description

Integration of the spectrum including uncertainty calculation.

Usage

```
signal_integrate(object, background, ...)

## S4 method for signature 'GammaSpectrum,missing'
signal_integrate(object, range = NULL, energy = FALSE)

## S4 method for signature 'GammaSpectrum,GammaSpectrum'
signal_integrate(object, background, range = NULL, energy = FALSE)

## S4 method for signature 'GammaSpectrum,numeric'
signal_integrate(object, background, range = NULL, energy = FALSE)

## S4 method for signature 'GammaSpectra,missing'
signal_integrate(object, range = NULL, energy = FALSE, simplify = TRUE)

## S4 method for signature 'GammaSpectra,GammaSpectrum'
signal_integrate(
```

```
  object,
  background,
  range = NULL,
  energy = FALSE,
  simplify = TRUE
)

## S4 method for signature 'GammaSpectra,numeric'
signal_integrate(
  object,
  background,
  range = NULL,
  energy = FALSE,
  simplify = TRUE
)
```

Arguments

object	A GammaSpectrum or GammaSpectra object.
background	A GammaSpectrum object.
...	Currently not used.
range	A length-two numeric vector giving the energy range to integrate within (in keV).
energy	A logical scalar: use the energy or count threshold for the signal integration
simplify	A logical scalar: should the result be simplified to a matrix? The default value, FALSE, returns a list.

Details

The function supports two integration techniques (see Guérin & Mercier 2011), the (1) count threshold integration and the (2) energy integration method:

The count integration technique (energy = FALSE) integrates all counts in given range:

$$A = \frac{\Sigma_i^N S_i}{t_{live}}$$

Contrary, the energy integration techniques is the integrated cross-product of counts and corresponding energy per channel:

$$A = \frac{\Sigma_i^N S_i \times E_i}{t_{live}}$$

A is the area, S_i is the signal in the i^{th} channel, N the number of channels, E_i the energy of the corresponding channel in keV. t_{live} is the live time of the measurement in s.

For calculating the uncertainties, Poisson statistics are assumed and hence the errors is calculated as:

$$\sigma_A = \frac{\sqrt{A}}{t_{live}}$$

Value

If `simplify` is `FALSE` (the default) returns a `list` of numeric vectors (the signal value and its error), else returns a `matrix`.

Note

The integration assumes that each spectrum has an energy scale.

Author(s)

N. Frerebeau

References

Guérin, G. & Mercier, M. (2011). Determining Gamma Dose Rates by Field Gamma Spectroscopy in Sedimentary Media: Results of Monte Carlo Simulations. *Radiation Measurements*, 46(2), p. 190-195. doi:10.1016/j.radmeas.2010.10.003.

Mercier, N. & Falguères, C. (2007). Field Gamma Dose-Rate Measurement with a NaI(Tl) Detector: Re-Evaluation of the "Threshold" Technique. *Ancient TL*, 25(1), p. 1-4.

See Also

Other signal processing: `baseline`, `peaks_find()`, `peaks_search()`, `signal_slice()`, `signal_split()`, `signal_stabilize()`, `smooth()`

---

signal_slice *Choose channels by Position*

Description

Choose channels by position.

Usage

```
signal_slice(object, ...)

## S4 method for signature 'GammaSpectrum'
signal_slice(object, ...)

## S4 method for signature 'GammaSpectra'
signal_slice(object, ...)
```

Arguments

`object`	A GammaSpectrum or GammaSpectra object.
`...`	`integer` values giving the channels of the spectrum to be kept/dropped (see below). Numeric values are coerced to integer as by `as.integer()` (and hence truncated towards zero).

Details

Either positive values to keep, or negative values to drop, should be provided. The values provided must be either all positive or all negative.

If no value is provided, an attempt is made to define the number of channels to skip at the beginning of the spectrum. This drops all channels before the highest count maximum. This is intended to deal with the artefact produced by the rapid growth of random background noise towards low energies.

Value

A GammaSpectrum or GammaSpectra object.

Author(s)

N. Frerebeau

See Also

Other signal processing: baseline, peaks_find(), peaks_search(), signal_integrate(), signal_split(), signal_stabilize(), smooth()

Examples

```
## Import CNF files
spc_file <- system.file("extdata/LaBr.CNF", package = "gamma")
spc <- read(spc_file)

## Plot spectrum
plot(spc)

## Slice
sliced <- signal_slice(spc)
plot(sliced)

sliced <- signal_slice(spc, -c(1:35))
plot(sliced)

sliced <- signal_slice(sliced, 450:550)
plot(sliced)

## Split
g <- rep(c("A", "B", "C"), c(250, 500, 274))
splited <- signal_split(spc, g)
plot(splited, facet = TRUE)
```

signal_split *Split*

Description

Split

Usage

```
signal_split(object, ...)

## S4 method for signature 'GammaSpectrum'
signal_split(object, groups)
```

Arguments

object	A GammaSpectrum object.
...	Currently not used.
groups	A a factor in the sense that as.factor(groups) defines the grouping (see split).

Value

A GammaSpectra object.

Author(s)

N. Frerebeau

See Also

Other signal processing: baseline, peaks_find(), peaks_search(), signal_integrate(), signal_slice(), signal_stabilize(), smooth()

Examples

```
## Import CNF files
spc_file <- system.file("extdata/LaBr.CNF", package = "gamma")
spc <- read(spc_file)

## Plot spectrum
plot(spc)

## Slice
sliced <- signal_slice(spc)
plot(sliced)

sliced <- signal_slice(spc, -c(1:35))
```

```
plot(sliced)

sliced <- signal_slice(sliced, 450:550)
plot(sliced)

## Split
g <- rep(c("A", "B", "C"), c(250, 500, 274))
splited <- signal_split(spc, g)
plot(splited, facet = TRUE)
```

`signal_stabilize` *Transform Intensities*

Description

Transform Intensities

Usage

```
signal_stabilize(object, ...)

## S4 method for signature 'GammaSpectrum'
signal_stabilize(object, f, ...)

## S4 method for signature 'GammaSpectra'
signal_stabilize(object, f, ...)
```

Arguments

`object`	A GammaSpectrum object.
`...`	Extra arguments to be passed to `f`.
`f`	A `function` that takes a numeric vector as argument and returns a numeric vector.

Details

The stabilization step aims at improving the identification of peaks with a low signal-to-noise ratio. This particularly targets higher energy peaks.

Value

A GammaSpectrum or GammaSpectra object with transformed intensities.

Author(s)

N. Frerebeau

See Also

Other signal processing: baseline, peaks_find(), peaks_search(), signal_integrate(), signal_slice(), signal_split(), smooth()

---

smooth *Smooth*

Description

Smoothes intensities.

Usage

```
signal_smooth(object, ...)

smooth_rectangular(object, ...)

smooth_triangular(object, ...)

smooth_savitzky(object, ...)

## S4 method for signature 'GammaSpectrum'
signal_smooth(object, method = c("rectangular", "triangular", "savitzky"), ...)

## S4 method for signature 'GammaSpectra'
signal_smooth(object, method = c("rectangular", "triangular", "savitzky"), ...)

## S4 method for signature 'GammaSpectrum'
smooth_rectangular(object, m = 3, ...)

## S4 method for signature 'GammaSpectra'
smooth_rectangular(object, m = 3, ...)

## S4 method for signature 'GammaSpectrum'
smooth_savitzky(object, m = 3, p = 2, ...)

## S4 method for signature 'GammaSpectra'
smooth_savitzky(object, m = 3, p = 2, ...)

## S4 method for signature 'GammaSpectrum'
smooth_triangular(object, m = 3, ...)

## S4 method for signature 'GammaSpectra'
smooth_triangular(object, m = 3, ...)
```

Arguments

`object`	A GammaSpectrum or GammaSpectra object.
`...`	Extra parameters to be passed to further methods.
`method`	A `character` string specifying the smoothing method to be used. It must be one of "`unweighted`" (default), "`weighted`" or "`savitzky`" (see details). Any unambiguous substring can be given.
`m`	An odd `integer` giving the number of adjacent points to be used.
`p`	An `integer` giving the polynomial degree. Only used if `method` is "`savitzky`".

Details

The following smoothing methods are available:

`rectangular` Unweighted sliding-average or rectangular smooth. It replaces each point in the signal with the average of m adjacent points.

`triangular` Weighted sliding-average or triangular smooth. It replaces each point in the signal with the weighted mean of m adjacent points.

`savitzky` Savitzky-Golay filter. This method is based on the least-squares fitting of polynomials to segments of m adjacent points.

There will be $(m-1)/2$ points both at the beginning and at the end of the spectrum for which a complete m-width smooth cannot be calculated. To prevent data loss, progressively smaller smooths are used at the ends of the spectrum if `method` is `unweighted` or `weighted`. If the Savitzky-Golay filter is used, the original $(m-1)/2$ points at the ends of the spectrum are preserved.

Value

A GammaSpectrum or GammaSpectra object.

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 signal processing: `baseline`, `peaks_find()`, `peaks_search()`, `signal_integrate()`, `signal_slice()`, `signal_split()`, `signal_stabilize()`

Examples

```
# Import CNF files
spc_file <- system.file("extdata/LaBr.CNF", package = "gamma")
spc <- read(spc_file)
spc <- signal_slice(spc, -c(1:35))

# Plot raw spectrum
spc_clean <- signal_correct(spc)
plot(spc_clean)

# Rectangular smooth
spc_unweighted <- smooth_rectangular(spc, m = 3)
spc_unweighted_clean <- signal_correct(spc_unweighted)
plot(spc_unweighted_clean)

# Triangular smooth
spc_weighted <- smooth_triangular(spc, m = 5)
spc_weighted_clean <- signal_correct(spc_weighted)
plot(spc_weighted_clean)

# Savitzky-Golay
spc_savitzky <- smooth_savitzky(spc, m = 21, p = 2)
spc_savitzky_clean <- signal_correct(spc_savitzky)
plot(spc_savitzky_clean)
```

---

subset *Extract or Replace Parts of an Object*

Description

Operators acting on objects to extract or replace parts.

Usage

```
## S4 method for signature 'GammaSpectrum'
x[[i]]

## S4 method for signature 'GammaSpectra'
x[i, j]

## S4 method for signature 'DoseRateModel'
x[[i]]

## S4 method for signature 'CalibrationCurve'
x[[i]]

## S4 method for signature 'PeakPosition'
x[[i]]
```

Arguments

x	An object from which to extract element(s) or in which to replace element(s).
i, j	Indices specifying elements to extract or replace. Indices are numeric, integer or character vectors or empty (missing) or NULL. Numeric values are coerced to integer as by as.integer() (and hence truncated towards zero). Character vectors will be matched to the name of the elements. An empty index (a comma separated blank) indicates that all entries in that dimension are selected.

Value

A subsetted object.

Author(s)

N. Frerebeau

See Also

Other mutator: mutator

---

summarise *Summarize*

Description

Summarize

Usage

```
summarise(object, ...)

## S4 method for signature 'GammaSpectrum'
summarise(object)

## S4 method for signature 'GammaSpectra'
summarise(object)

## S4 method for signature 'DoseRateModel'
summarise(object)

## S4 method for signature 'CalibrationCurve'
summarise(object)
```

Arguments

object	A GammaSpectrum or GammaSpectra object.
...	Currently not used.

Value

A data.frame.

Author(s)

N. Frerebeau

See Also

Other IO: read()

Examples

```
## Import a Canberra CNF file
cnf_file <- system.file("extdata/LaBr.CNF", package = "gamma")
spc <- read(cnf_file)
summarise(spc)

## Import all CNF files in a given directory
spc_dir <- system.file("extdata/BDX_LaBr_1/calibration", package = "gamma")
spc <- read(spc_dir)
summarise(spc)
```

Index

∗ **IO**
  read, 32
  summarise, 42
∗ **class**
  Baseline-class, 6
  CalibrationCurve-class, 8
  coerce, 10
  GammaSpectra-class, 18
  GammaSpectrum-class, 19
  PeakPosition-class, 26
∗ **datasets**
  AIX_NaI_1, 3
  BDX_LaBr_1, 7
  clermont, 9
  clermont_2024, 10
∗ **dose rate**
  doserate, 11
∗ **energy**
  energy, 15
∗ **mutator**
  mutator, 21
  subset, 41
∗ **operator**
  operator, 25
∗ **plot**
  plot, 30
∗ **signal processing**
  baseline, 3
  peaks_find, 27
  peaks_search, 29
  signal_integrate, 33
  signal_slice, 35
  signal_split, 37
  signal_stabilize, 38
  smooth, 39
.Baseline *(Baseline-class)*, 6
.CalibrationCurve *(CalibrationCurve-class)*, 8
.DoseRateModel *(CalibrationCurve-class)*, 8
.GammaSpectra *(GammaSpectra-class)*, 18
.GammaSpectrum *(GammaSpectrum-class)*, 19
.PeakPosition *(PeakPosition-class)*, 26
[,GammaSpectra-method *(subset)*, 41
[[,CalibrationCurve-method *(subset)*, 41
[[,DoseRateModel-method *(subset)*, 41
[[,GammaSpectrum-method *(subset)*, 41
[[,PeakPosition-method *(subset)*, 41

AIX_NaI_1, 3, *8–10*
Arith,GammaSpectrum,GammaSpectrum-method *(operator)*, 25
Arith,GammaSpectrum,numeric-method *(operator)*, 25
as.integer(), *20*, *26*, *35*, *42*
as.matrix.GammaSpectrum *(coerce)*, 10

BaseLine, *5*
baseline, 3, *28*, *29*, *35–37*, *39*, *40*
BaseLine-class *(Baseline-class)*, 6
Baseline-class, 6
baseline-method *(baseline)*, 3
baseline_linear *(baseline)*, 3
baseline_linear,GammaSpectra-method *(baseline)*, 3
baseline_linear,GammaSpectrum-method *(baseline)*, 3
baseline_linear-method *(baseline)*, 3
baseline_rubberband *(baseline)*, 3
baseline_rubberband,GammaSpectra-method *(baseline)*, 3
baseline_rubberband,GammaSpectrum-method *(baseline)*, 3
baseline_rubberband-method *(baseline)*, 3
baseline_snip *(baseline)*, 3
baseline_snip,GammaSpectra-method *(baseline)*, 3
baseline_snip,GammaSpectrum-method *(baseline)*, 3

baseline_snip-method *(baseline)*, 3
BDX_LaBr_1, *3*, 7, *9, 10*

CalibrationCurve, *3*, *8*, *12, 13*
CalibrationCurve-class, 8
character, *4*, *13*, *19*, *26*, *28*, *31, 32*, *40*, *42*
clermont, *3*, *8*, 9, *10*
clermont_2024, *3*, *8, 9*, 10
coerce, *7*, *9*, 10, *19, 20*, *27*
Compare,GammaSpectrum,GammaSpectrum-method *(operator)*, 25
Compare,GammaSpectrum,numeric-method *(operator)*, 25

data.frame, *8–10*, *13*, *20*, *27*, *43*
dose_fit *(doserate)*, 11
dose_fit,GammaSpectra,data.frame-method *(doserate)*, 11
dose_fit,GammaSpectra,GammaSpectrum,matrix-method *(doserate)*, 11
dose_fit,GammaSpectra,GammaSpectrumOrNumeric,data.frame-method *(doserate)*, 11
dose_fit,GammaSpectra,GammaSpectrumOrNumeric,matrix-method *(doserate)*, 11
dose_fit-method *(doserate)*, 11
dose_predict *(doserate)*, 11
dose_predict,CalibrationCurve,GammaSpectra-method *(doserate)*, 11
dose_predict,CalibrationCurve,GammaSpectrum-method *(doserate)*, 11
dose_predict,CalibrationCurve,missing-method *(doserate)*, 11
dose_predict-method *(doserate)*, 11
doserate, 11
DoseRateModel, *8*
DoseRateModel-class *(CalibrationCurve-class)*, 8

energy, 15
energy_calibrate *(energy)*, 15
energy_calibrate(), *20*
energy_calibrate,GammaSpectra,CalibrationCurve-method *(energy)*, 15
energy_calibrate,GammaSpectra,GammaSpectrum-method *(energy)*, 15
energy_calibrate,GammaSpectra,list-method *(energy)*, 15
energy_calibrate,GammaSpectra,lm-method *(energy)*, 15
energy_calibrate,GammaSpectra,PeakPosition-method *(energy)*, 15
energy_calibrate,GammaSpectrum,CalibrationCurve-method *(energy)*, 15
energy_calibrate,GammaSpectrum,GammaSpectrum-method *(energy)*, 15
energy_calibrate,GammaSpectrum,list-method *(energy)*, 15
energy_calibrate,GammaSpectrum,lm-method *(energy)*, 15
energy_calibrate,GammaSpectrum,PeakPosition-method *(energy)*, 15
energy_calibrate-method *(energy)*, 15

factor, *37*
function, *38*

GammaSpectra, *4, 5*, *12, 13*, *17*, *32*, *34–38*, *40*, *42*
GammaSpectra-class, 18
GammaSpectrum, *4–6*, *12, 13*, *17, 18*, *25, 26*, *28, 29*, *32*, *34–38*, *40*, *42*
GammaSpectrum-class, 19
get *(mutator)*, 21
get_channels *(mutator)*, 21
get_channels,GammaSpectra-method *(mutator)*, 21
get_channels,GammaSpectrum-method *(mutator)*, 21
get_channels,PeakPosition-method *(mutator)*, 21
get_channels-method *(mutator)*, 21
get_counts *(mutator)*, 21
get_counts,GammaSpectra-method *(mutator)*, 21
get_counts,GammaSpectrum-method *(mutator)*, 21
get_counts-method *(mutator)*, 21
get_energy *(mutator)*, 21
get_energy,GammaSpectra-method *(mutator)*, 21
get_energy,GammaSpectrum-method *(mutator)*, 21
get_energy,PeakPosition-method *(mutator)*, 21
get_energy-method *(mutator)*, 21
get_energy_calibration *(mutator)*, 21
get_energy_calibration,GammaSpectra-method *(mutator)*, 21

get_energy_calibration,GammaSpectrum-method (mutator), 21
get_energy_calibration-method (mutator), 21
get_hash (mutator), 21
get_hash,GammaSpectra-method (mutator), 21
get_hash,GammaSpectrum-method (mutator), 21
get_hash,PeakPosition-method (mutator), 21
get_hash-method (mutator), 21
get_livetime (mutator), 21
get_livetime,GammaSpectra-method (mutator), 21
get_livetime,GammaSpectrum-method (mutator), 21
get_livetime-method (mutator), 21
get_method (mutator), 21
get_method,Baseline-method (mutator), 21
get_method-method (mutator), 21
get_names (mutator), 21
get_names,GammaSpectra-method (mutator), 21
get_names,GammaSpectrum-method (mutator), 21
get_names-method (mutator), 21
get_rates (mutator), 21
get_rates,GammaSpectra-method (mutator), 21
get_rates,GammaSpectrum-method (mutator), 21
get_rates-method (mutator), 21
get_realtime (mutator), 21
get_realtime,GammaSpectra-method (mutator), 21
get_realtime,GammaSpectrum-method (mutator), 21
get_realtime-method (mutator), 21
get_residuals (mutator), 21
get_residuals,DoseRateModel-method (mutator), 21
get_residuals-method (mutator), 21
ggplot2::ggplot, *31*

has_calibration (energy), 15
has_calibration,CalibrationCurve-method (energy), 15
has_calibration,GammaSpectra-method (energy), 15
has_calibration,GammaSpectrum-method (energy), 15
has_energy (energy), 15
has_energy,GammaSpectra-method (energy), 15
has_energy,GammaSpectrum-method (energy), 15
hyperSpec::spc.rubberband(), *5*

integer, *5*, *20*, *26*, *28*, *29*, *31*, *35*, *40*, *42*
IsoplotR::ellipse(), *31*
IsoplotR::isochron(), *31*

length,GammaSpectrum-method (mutator), 21
linear model, *20*
list, *8*, *13*, *17*, *18*, *34*, *35*
Logic,GammaSpectrum,GammaSpectrum-method (operator), 25
Logic,GammaSpectrum,logical-method (operator), 25
Logic,GammaSpectrum,numeric-method (operator), 25
logical, *5*, *13*, *17*, *24*, *25*, *31*, *34*

Math,GammaSpectrum-method (operator), 25
Math2,GammaSpectrum-method (operator), 25
matrix, *13*, *14*, *20*, *27*, *34*, *35*
mutator, 21, *42*

NA, *24*
numeric, *5*, *8*, *12–14*, *19*, *20*, *25*, *26*, *29*, *31*, *34*, *42*

operator, 25

PeakPosition, *17*, *28*, *29*
PeakPosition-class, 26
peaks_find, *6*, 27, *29*, *35–37*, *39*, *40*
peaks_find,GammaSpectrum-method (peaks_find), 27
peaks_find-method (peaks_find), 27
peaks_search, *6*, *28*, 29, *35–37*, *39*, *40*
peaks_search,GammaSpectrum,integer-method (peaks_search), 29
peaks_search,GammaSpectrum,numeric-method (peaks_search), 29

peaks_search-method (peaks_search), 29
plot, 30
plot,CalibrationCurve,missing-method (plot), 30
plot,GammaSpectra,missing-method (plot), 30
plot,GammaSpectrum,Baseline-method (plot), 30
plot,GammaSpectrum,missing-method (plot), 30
plot,GammaSpectrum,PeakPosition-method (plot), 30
plot-method (plot), 30
POSIXct, *19*

range_channels (mutator), 21
range_channels,GammaSpectra-method (mutator), 21
range_channels,GammaSpectrum-method (mutator), 21
range_channels-method (mutator), 21
range_energy (mutator), 21
range_energy,GammaSpectra-method (mutator), 21
range_energy,GammaSpectrum-method (mutator), 21
range_energy-method (mutator), 21
read, 32, *43*
read,character-method (read), 32
read-method (read), 32
round(), *25*
rxylib::read_xyData(), *32, 33*

set (mutator), 21
set_energy,PeakPosition,numeric-method (mutator), 21
set_energy-method (mutator), 21
set_energy<- (mutator), 21
set_energy<-,PeakPosition,numeric-method (mutator), 21
set_energy_calibration,GammaSpectra-method (mutator), 21
set_energy_calibration,GammaSpectrum-method (mutator), 21
set_energy_calibration-method (mutator), 21
set_energy_calibration<- (mutator), 21
set_energy_calibration<-,GammaSpectra-method (mutator), 21
set_energy_calibration<-,GammaSpectrum-method (mutator), 21
set_method,Baseline-method (mutator), 21
set_method-method (mutator), 21
set_method<- (mutator), 21
set_method<-,Baseline-method (mutator), 21
set_name,GammaSpectra-method (mutator), 21
set_names,GammaSpectrum-method (mutator), 21
set_names-method (mutator), 21
set_names<- (mutator), 21
set_names<-,GammaSpectra-method (mutator), 21
set_names<-,GammaSpectrum-method (mutator), 21
signal_baseline (baseline), 3
signal_baseline,GammaSpectra-method (baseline), 3
signal_baseline,GammaSpectrum-method (baseline), 3
signal_correct (baseline), 3
signal_correct,GammaSpectra-method (baseline), 3
signal_correct,GammaSpectrum-method (baseline), 3
signal_correct-method (baseline), 3
signal_integrate, *6*, *28, 29*, 33, *36, 37*, *39, 40*
signal_integrate(), *13–15*
signal_integrate,GammaSpectra,GammaSpectrum-method (signal_integrate), 33
signal_integrate,GammaSpectra,missing-method (signal_integrate), 33
signal_integrate,GammaSpectra,numeric-method (signal_integrate), 33
signal_integrate,GammaSpectrum,GammaSpectrum-method (signal_integrate), 33
signal_integrate,GammaSpectrum,missing-method (signal_integrate), 33
signal_integrate,GammaSpectrum,numeric-method (signal_integrate), 33
signal_integrate-method (signal_integrate), 33
signal_slice, *6*, *28, 29*, *35*, 35, *37*, *39, 40*
signal_slice,GammaSpectra-method (signal_slice), 35

signal_slice,GammaSpectrum-method
(signal_slice), 35
signal_slice-method (signal_slice), 35
signal_smooth (smooth), 39
signal_smooth,GammaSpectra-method
(smooth), 39
signal_smooth,GammaSpectrum-method
(smooth), 39
signal_smooth-method (smooth), 39
signal_split, *6*, *28, 29*, *35, 36*, 37, *39, 40*
signal_split,GammaSpectrum-method
(signal_split), 37
signal_split-method (signal_split), 37
signal_stabilize, *6*, *28, 29*, *35–37*, 38, *40*
signal_stabilize,GammaSpectra-method
(signal_stabilize), 38
signal_stabilize,GammaSpectrum-method
(signal_stabilize), 38
signal_stabilize-method
(signal_stabilize), 38
signif(), *25*
smooth, *6*, *28, 29*, *35–37*, *39*, 39
smooth_rectangular (smooth), 39
smooth_rectangular,GammaSpectra-method
(smooth), 39
smooth_rectangular,GammaSpectrum-method
(smooth), 39
smooth_rectangular-method (smooth), 39
smooth_savitzky (smooth), 39
smooth_savitzky,GammaSpectra-method
(smooth), 39
smooth_savitzky,GammaSpectrum-method
(smooth), 39
smooth_savitzky-method (smooth), 39
smooth_triangular (smooth), 39
smooth_triangular,GammaSpectra-method
(smooth), 39
smooth_triangular,GammaSpectrum-method
(smooth), 39
smooth_triangular-method (smooth), 39
split, *37*
stats::lm, *17*
subset, *24*, 41
summarise, *33*, 42
summarise,CalibrationCurve-method
(summarise), 42
summarise,DoseRateModel-method
(summarise), 42
summarise,GammaSpectra-method
(summarise), 42
summarise,GammaSpectrum-method
(summarise), 42
summarise-method (summarise), 42
Summary,GammaSpectrum-method
(operator), 25