Package 'BBEST'

January 20, 2025

Author Anton Gagin and Igor Levin with contributions from Charles R. Hogg III

Maintainer Anton Gagin <av.gagin@gmail.com>

Version 0.1-8

Type Package

Title Bayesian Estimation of Incoherent Neutron Scattering Backgrounds

Description We implemented a Bayesian-statistics approach for subtraction of incoherent scattering from neutron total-scattering data. In this approach, the estimated background signal associated with incoherent scattering maximizes the posterior probability, which combines the likelihood of this signal in reciprocal and real spaces with the prior that favors smooth lines. The description of the corresponding approach could be found at Gagin and Levin (2014) <DOI:10.1107/S1600576714023796>.

Date 2020-11-18

License GPL-3

Imports DEoptim, aws, grid, ggplot2, reshape2, shiny, methods

LazyData yes

NeedsCompilation no

Repository CRAN

RoxygenNote 7.1.1

Encoding UTF-8

Date/Publication 2020-11-19 13:20:05 UTC

Contents

`BBEST-package` *Bayesian Background Estimation.*

Description

In this package we implemented a Bayesian-statistics approach for subtraction of incoherent scattering from neutron total-scattering data. In this approach, the estimated background signal associated with incoherent scattering maximizes the posterior probability, which combines the likelihood of this signal in reciprocal and real spaces with the prior that favors smooth lines.

To cite the BBEST package type: 'citation("BBEST")' (without the single quotes).

For a listing of all routines in the BBEST package type: 'library(help="BBEST")'

To start the Graphical User Interface type: 'runUI()'

To start a simple command-line guide type: 'guide()'

Details

Package:	BBEST
Type:	Package
Version:	0.1-0
Date:	2014-08-11
License:	GPL-3

Author(s)

Anton Gagin and Igor Levin with contributions from Charles R. Hogg III
Maintainer: Anton Gagin <anton.gagin@nist.gov>

References

BBEST-package

`calc.Gr` *Calculate and plot the Pair Distribution Function*

Description

Calculates and plots the corrected Pair Distribution Function.

Usage

```
calc.Gr(fit.results, rho.0, plot=TRUE, r.min = 0, r.max = 5,
        dr = 0.01, Q.min = NA, Q.max = NA, nsd = 2, gr.compare=NA)
```

Arguments

`fit.results`	the return value of `do.fit`.
`plot`	logical, whether to plot the PDF.
`rho.0`	numeric, the atomic number density of the material: the number of atoms per unit cell divided by a volume of the unit cell.
`r.min`, `r.max`, `dr`	numerics. Function is plotted in the region `[r.min, r.max]`.
`Q.min`, `Q.max`	numerics. To calculate the sine-Fourier transform, the total scattering function S(Q) is "terminated" at a certain `Q=Qmax` point. The best Qmax point to terminate S(Q) (that corresponds to the value of S(Q)-1 closest to zero) is sought in the `[Q.min,Q.max]` region.
`nsd`	numeric, the number of standard deviations to plot the uncertainty.
`gr.compare`	numeric vector. If not `NA`, specifies the function to add to the plot. Should correspond to the same grid (`[r.min, r.max, dr]`).

Details

The function uses `ggplot2` package for plotting. `ggplot2` package can be installed by typing `install.packges("ggplot2")`.

Value

A list with elements:

r	numeric vector of grid points
gr	numeric vector, indicates the corrected Pair Distribution Function.
stdev	numeric vector, indicates estimated standard deviation.

See Also

do.fit

---

do.fit *Estimate background*

Description

do.fit estimates the background using the Bayesian approach and Differential Evolution algorithm.

Usage

```
do.fit(data, bounds.lower, bounds.upper, scale=c(1,1), knots.x=NA,
       knots.n=NA, analytical=FALSE, stdev=TRUE, control=list(), p.bkg=.5,
   save.to="")
```

Arguments

data	an object of type data. See set.data for details.
bounds.lower, bounds.upper	numerics specifying the lower and upper bounds for the fitted spline values.
scale	numeric vector which, if applicable, determines the bounds for the fitted scale parameter. The default value of c(1,1) means a no-scale fit. See details.
knots.x	numeric vector which, if not NA, specifies the knot positions.
knots.n	numeric, the number of knots. If knots.x is NA then knots.n equidistant knots will be created.
analytical	logical. If TRUE background is approximated by an analytical function $f(x) = P_1 \exp(-P_2 x) x^{P_3} + P_4/[(x - P_5)^2 + P_6^2]$.
stdev	logical, whether to calculate the uncertainty for the estimated background. Should be set to FALSE if analytical=TRUE.
control	list, the return value of set.control. Specifies various parameters of the Differential Evolution optimization algorithm implemented in DEoptim.
p.bkg	numeric, the probability that a single pixel contains "only" a background.
save.to	character, a filename for saving the results.

Details

If information on the low-r behavior of G(r) is provided, the global intensity scale and atomic displacement parameters can be fitted along with the positions of the knots, (`set.Gr`). To fit normalization parameter set bounds in `scale` for the desired values. To fit Atomic Displacement Parameters see `set.SB`.

In most cases `p.bkg` should be set to its default value 0.5.

For further details see `BBEST-package`.

Value

A list with elements:

`x`	numeric vector of grid points
`curves`	list, see below.
`uncrt`	list, see below.
`knots`	list with elements `x` and `y` that specify the positions of the knots and the corresponding fitted intensity values, respectively.
`pars`	numeric vector. If the background is approximated using the analytical function, contains all the relevant parameters `P`.
`scale`	fitted value of the `scale` parameter, if used.
`ADP`	fitted values of the atomic displacement parameters, if applicable.
`fit.details`	list, see below.

Element `curves` is a list with sub-elements:

`y`	numeric vector of the (normalized) function values.
`bkg`	numeric vector, the estimated background.
`SB`	numeric vector, the (fitted) coherent baseline.

Element `uncrt` is a list with sub-elements:

`stdev`	numeric vector, indicates estimated standard deviations for the reconstructed signal.
`stdev.r`	numeric vector, indicates estimated standard deviations for a reconstructed signal in r-space.
`hess`	Hessian matrix for a $\psi(c)$ function.
`cov.matrix`	covariance matrix, i.e. the inverse of the Hessian.
`cov.matrix.r`	covariance matrix in r-space.

Element `fit.details` is a list with sub-elements:

`lambda`	numeric vector, the estimated mean magnitude of the signal.
`sigma`	numeric vector, the estimated Gaussian noise.
`knots.n`	the number of knots used in the fit.
`knots.x`	knot positions used in the fit.

control	see the control argument.
Gr	list contacting information on the low-r behaviour of G(r) . See set.Gr for details.
n.atoms	numeric vector, number of different atoms per unit cell.
scatter.length	numeric vector, atomic scattering factors.

References

Ardia, D., Mullen, K., Peterson, B. & Ulrich, J. (2011): DEoptim. R Package Version 2.2-2. https://CRAN.R-project.org/package=DEoptim.

Mullen, K.M., Ardia, D., Gil, D., Windover, D., Cline, J. (2011): DEoptim: An R Package for Global Optimization by Differential Evolution. *J. Stat. Softw.*, **40**(6), 1-26. https://www.jstatsoft.org/article/view/v040i06.

---

do.fit.banks *Estimate the background for individual banks*

Description

do.fit estimates the background for individual banks according to the Bayesian approach using the Differential Evolution algorithm

Usage

```
do.fit.banks(data, bounds.lower, bounds.upper, knots.n.left,
             knots.n.right, x.boundary, analytical=FALSE, control,
             save.to="")
```

Arguments

data	an object of type data. See set.data for details.
bounds.lower, bounds.upper	numerics, lower and upper bounds for the fitted spline values.
knots.n.left, knots.n.right, x.boundary	numerics that specify the number of knots. knots.n.left and knots.n.right knots are created on the left and on the right of x.boundary point, respectively.
analytical	logical. If TRUE background is approximated by an analytical function $f(x) = P_1 \exp(-P_2 x) x^{P_3} + P_4/[(x - P_5)^2 + P_6^2]$.
control	list, the return value of set.control. Specifies various parameters of the Differential Evolution optimization algorithm implemented in DEoptim.
save.to	character, a filename for saving the results.

Details

This function simplifies the procedure for estimating the background for several detector banks by a multiple call of do.fit. Other relevant parameters are set to: stdev=FALSE, scale=NA, p.bkg=.5.

For neutron scattering, the incoherent background exhibits a broad peak at low Q and decays gradually at higher Q. Hence, we suggest to use different numbers of knots for the low- and high-Q regions. See BBEST-package for details.

Value

A list of elements. Each element contains a return value of do.fit for the corresponding data bank.

See Also

do.fit, BBEST-package

---

do.iter *Estimate the background*

Description

do.iter performs adaptive Bayesian estimation of the background.

Usage

```
do.iter(fit.results, local = TRUE, eps = 1e-04,
        n.iter = 10000, save.to = "")
```

Arguments

fit.results	list. The return value of do.fit.
local	logical. If TRUE, gradient descent method is used to find background estimation. If FALSE, Differential Evolution is used.
eps	numeric, the desired accuracy for spline values.
n.iter	numeric, number of iterations for a gradient descent method, see details.
save.to	character, the filename for saving the results.

Details

An adaptation of neutron scattering data for a Bayesian background separation procedure. The method is detailed elsewhere*.

First, use the function do.fit to estimate the background *from* the low-r information in G(r). do.iter procedure estimates the background *without* low-r information, calculates the difference between the two estimates, subtracts this difference from the scattering data and finds the new estimate of the background.

Value

An object `fit.results` with modified elements `fit.results$curves$bkg`, `fit.results$curves$y` and `fit.results$curves$corr`. See `do.fit` for details.

References

*Gagin, A. and Levin, I. Hydrogen background estimation in neutron total scattering experiments. Submitted for publication.

`fix.merge` *Merge .fix files*

Description

`fix.merge` merges several .fix files into a specified file in a form suitable for *PDFgetN*.

Usage

```
fix.merge(outfile, infile1, infile2, ...)
```

Arguments

`outfile`	character, the filename for saving the data.
`infile1, infile2, ...`	files to merge.

See Also

`write.fix`, `read.sqa`, `do.fit.banks`, `BBEST-package`

`guide` *BBEST guide*

Description

`guide` is a function that guides through the Bayesian procedure for estimating the background

Usage

```
guide()
```

Value

A list with elements:

`fit.res`	the return value of `do.fit`.
`data`	an object of type `data`, see `set.data`.
`gr`	the return value of `calc.Gr`.

mPlot.results *Plot the background estimate*

Description

Plots the estimated background and the corrected function.

Usage

```
mPlot.results(fit.results, label.x = "x", label.y = "y",
              xlim=NA, ylim=NA)
```

Arguments

fit.results	the return value of do.fit.
label.x, label.y	characters, titles for x and y axes.
xlim, ylim	numeric vectors with two entries. If not NA, specify x- and y-axis limits.

Details

The function uses ggplot2 and gridExtra packages for plotting. Packages can be installed by typing install.packges("ggplot2") and install.packges("gridExtra").

See Also

do.fit

mPlot.results.banks *Plot the background estimate for individual banks*

Description

Plots the background estimate for individual detector banks.

Usage

```
mPlot.results.banks(fit.results, label.x = "x", label.y = "y",
                    xlim=NA, ylim=NA)
```

Arguments

fit.results	the return value of do.fit.banks.
label.x, label.y	characters, titles for x and y axes.
xlim, ylim	numeric matrices of size (NB, 2), where NB is the number of data banks. If not NA, specify x- and y-axis limits.

See Also

do.fit.banks

---

mPlot.sqa *Plot the total normalized scattering intensity function* S(Q) *for individual detector banks*

Description

The function plots the total scattering functions S(Q) returned by PDFgetN in read.sqa.

Usage

```
mPlot.sqa(data)
```

Arguments

data	list, the return value of read.sqa.

See Also

read.sqa

---

prepare.banks.data *Prepare data for estimating the background*

Description

prepare.banks.data sets all the fit parameters, such as sigma, lambda and SB for a set of detector banks.

Usage

```
prepare.banks.data(data, n.banks=4, lambda_1, lambda_2, lambda_0,
                   x_1, x_2, n.atoms, scatter.length, ADP, n.regions)
```

Arguments

data	list of objects of type data. See read.sqa and set.data for details.
n.banks	numeric, number of banks.
lambda_1, lambda_2, lambda_0, x_1, x_2	parameters to be passed to set.lambda.
n.atoms, scatter.length, ADP	parameters to be passed to set.SB.
n.regions	parameter to be passed to set.sigma.

Details

This function simplifies setting the fit parameters for a set of detector banks by a multiple call of `set.sigma`, `set.SB`, and `set.lambda`.

Value

A list of objects of type `data` suitable for `do.fit.banks`.

See Also

`set.sigma`, `set.SB`, `set.lambda`

Progress-class *Reporting progress (object-oriented API)*

Description

Reports progress to the user during long-running operations.

Arguments

`session`	The Shiny session object, as provided by `shinyServer` to the server function.
`min`	The value that represents the starting point of the progress bar. Must be less than `max`.
`max`	The value that represents the end of the progress bar. Must be greater than `min`.
`message`	A single-element character vector; the message to be displayed to the user, or `NULL` to hide the current message (if any).
`detail`	A single-element character vector; the detail message to be displayed to the user, or `NULL` to hide the current detail message (if any). The detail message will be shown with a de-emphasized appearance relative to `message`.
`value`	Single-element numeric vector; the value at which to set the progress bar, relative to `min` and `max`. `NULL` hides the progress bar, if it is currently visible.

Details

This package exposes two distinct programming APIs for working with progress. `withProgress` and `setProgress` together provide a simple function-based interface, while the `Progress` reference class provides an object-oriented API.

Instantiating a `Progress` object causes a progress panel to be created, and it will be displayed the first time the `set` method is called. Calling `close` will cause the progress panel to be removed.

Methods

`initialize(session, min = 0, max = 1)` Creates a new progress panel (but does not display it).

`set(message = NULL, detail = NULL, value = NULL)` Updates the progress panel. When called the first time, the progress panel is displayed.

`close()` Removes the progress panel. Future calls to `set` and `close` will be ignored.

See Also

progressInit, withProgress

Examples

```
## Not run:
# server.R
shinyServer(function(input, output, session) {
  output$plot <- renderPlot({
    progress <- Progress$new(session, min=1, max=15)
    on.exit(progress$close())

    progress$set(message = 'Calculation in progress',
                 detail = 'This may take a while...')

    for (i in 1:15) {
      progress$set(value = i)
      Sys.sleep(0.5)
    }
    plot(cars)
  })
})

## End(Not run)
```

---

progressInit *Initialize progress*

Description

Call this function in your shinyUI definition if you intend to use progress in server.R.

Usage

```
progressInit()
```

See Also

withProgress, Progress

read.data *Read data from file*

Description

Reads data from a text file with columns "x", "y", and, optionally, "lambda", "sigma" and "SB".

Usage

```
read.data(file = stop("'file' must be specified"), ...)
```

Arguments

file	character, the name of the file which the data are to be read from.
...	further arguments to be passed to read.table (optional).

Details

This function implements one of the ways to load experimental data. The file must consist of a header with column names and several columns below. First two columns in file must be x and y values. The others could specify lambda, sigma and SB.

Value

An object of type data. See set.data for details.

read.sqa *Read data from a .sqa-file*

Description

This function reads .sqa-files generated by *PDFgetN*, which contain corrected total-scattering functions bank by bank.

Usage

```
read.sqa(file = stop("'file' must be specified"))
```

Arguments

file	character, the name of the file which the data are to be read from.

Value

List those elements are objects of type data. See set.data for details.

References

Peterson, P.F., Gutmann, M., Proffen, TH. & Billinge, S.J.L. (2000): PDFgetN: A User-Friendly Program to Extract the Total Scattering Structure Function and the Pair Distribution Function from Neutron Powder Diffraction Data. *J. Appl. Cryst.*, **33**, 1192. https://web.pa.msu.edu/cmp/billinge-group/programs/pdfgetn/pdfgetn_jac.pdf.

Proffen, TH., Peterson, P.F., Gutmann, M. & Billinge, S.J.L. (2009): PDFgetN Users Guide Version 1.6.6. http://pdfgetn.sourceforge.net/.

See Also

mPlot.sqa

---

read.sqb *Read data from a* .sqb*-file*

Description

This function reads .sqb-files generated by *PDFgetN*, which contain the corrected and blended total-scattering function S(Q).

Usage

```
read.sqb(file = stop("'file' must be specified"))
```

Arguments

file	character, the name of the file which the data are to be read from.

Value

An object of type data. See set.data for details.

References

Peterson, P.F., Gutmann, M., Proffen, TH. & Billinge, S.J.L. (2000): PDFgetN: A User-Friendly Program to Extract the Total Scattering Structure Function and the Pair Distribution Function from Neutron Powder Diffraction Data. *J. Appl. Cryst.*, **33**, 1192. https://web.pa.msu.edu/cmp/billinge-group/programs/pdfgetn/pdfgetn_jac.pdf.

Proffen, TH., Peterson, P.F., Gutmann, M. & Billinge, S.J.L. (2009): PDFgetN Users Guide Version 1.6.6. http://pdfgetn.sourceforge.net/.

runUI *Start the GUI*

Description

Starts the application and opens up the default web browser to view it.

Usage

```
runUI()
```

Details

Runs a **Shiny** application. This function normally does not return; interrupt boldR to stop the application (usually by pressing Ctrl+C or Esc).

set.control *Set controls for the Differential Evolution Algorithm*

Description

Specifies various parameters of the Differential Evolution optimization algorithm implemented in `DEoptim`.

Usage

```
set.control(CR=.85, F=.7, NP=300, itermax=2000, parallelType=1)
```

Arguments

`CR`	numeric, crossover probability from interval [0,1].
`F`	numeric, differential weighting factor from interval [0,2].
`NP`	numeric, number of population members
`itermax`	numeric, the number of iterations
`parallelType`	numeric, defines the type of parallelization to employ. `0` for a single-core run. If `parallelType=1` the program will use all the available cores, via the `parallel` package.

Details

For the most tasks, it is best to set `NP` to at least 10-15 times the length of the parameter vector.

Value

a list of elements suitable for `do.fit` and `do.fit.banks`.

References

Mullen, K.M., Ardia, D., Gil, D., Windover, D., Cline, J. (2011): DEoptim: An R Package for Global Optimization by Differential Evolution. *J. Stat. Softw.*, **40**(6), 1-26. https://www.jstatsoft.org/article/view/v040i06.

`set.data` *Set data*

Description

The function sets key parameters necessary for the fit, such as `sigma`, `lambda` and `SB`

Usage

```
set.data(x, y, sigma=NA, lambda=NA, SB=NA)
```

Arguments

`x`	numeric vector, specifies grid points.
`y`	numeric vector, specifies function values.
`sigma`	numeric vector, if not `NA`, specifies estimated noise.
`lambda`	numeric vector, if not `NA`, specifies estimated mean signal magnitude.
`SB`	numeric vector, if not `NA`, specifies estimated coherent baseline.

Details

One way (not the simplest) to prepare experimental data for the fit. This function returns a list of the above parameters – an object of type `data`. Objects of that type are used as arguments for some functions implemented in the package. In most cases only the elements `x` and `y` are required in the object `data`. However, all 5 elements (and one optional, see `set.Gr`) must be specified to execute the fit, i.e. prior to the `do.fit` call.

The object of that type can also be created via `read.data`, `read.sqa` and `read.sqb`. Parameters "sigma", "lamdba" and "SB" can be determined automatically, see `set data` keyword.

The general recipe for setting an object `data` is the following. If vectors `x` and `y` are stored in the text file, use `read.data`. If they are stored in a `.sqb`-file, call `read.sqb`. If they are stored in the memory, use `set.data`. Then use functions `set.sigma`, `set.lambda`, and `set.SB`) to specify the remaining parameters.

Value

A list with elements

`x`	numeric vector, specifies gridpoints.
`y`	numeric vector, specifies function values.
`sigma`	numeric vector, specifies estimated noise.
`lambda`	numeric vector, specifies estimated mean signal magnitude.
`SB`	numeric vector, specifies estimated coherent baseline.

`set.Gr` *Add information on the low-r behaviour of G(r)*

Description

Function to incorporate information on the low-r behaviour of G(r) into the Bayesian model.

Usage

```
set.Gr(data, r1=seq(0, 1, 0.005), r2=NA, rho.0,
       type1="gaussianNoise", type2=NA, sigma.f=NA, l=NA)
```

Arguments

`data`	an object of type `data`. See `set.data` for details.
`r1`, `r2`	numeric vectors, specify grids on which the G(r) behaviour is controlled.
`rho.0`	numeric, atomic number density of the material: a number of atoms per unit cell divided by a volume of the unit cell.
`type1`, `type2`	characters, specify the way to control the behavior of G(r). See details.
`sigma.f`, `l`	numerics or numeric vectors, specify parameters for a squared-exponential covariance function.

Details

`type1` can be either "gaussianNoise" or "correlatedNoise". G(r) is restricted to the $-4\pi\rho.0r1$ line plus independent Gaussian noise or correlated Gaussian noise, respectively.

`type2` can be either "secondDeriv" or "gaussianProcess" to impose smoothness conditions over the interval `r2`. If `type2` is "secondDeriv", a minimum of the second derivative is sought. If `type2` is "gaussianProcess", the smoothness is controlled via the Gaussian process using parameters sigma.f and l.

According to our experience, the most efficient way is to impose `type1="gaussianNoise"` and `type2=NA` conditions.

Value

An object of type `data`.

set.lambda *Set mean signal magnitude*

Description

set.lambda sets the mean height of the peaks over region x.

Usage

```
set.lambda(data, lambda=NA, lambda_1=NA, lambda_2=NA,
           lambda_0=NA, x_1=NA, x_2=NA)
```

Arguments

data	an object of type data. See set.data for details.
lambda	numeric vector. If not NA, specifies (approximate) the mean magnitude of the signal. This estimate does not need to be accurate. lambda can be estimated as a smooth function that crosses centres of the signal peaks.
lambda_1, lambda_2, lambda_0, x_1, x_2	numerics. If lambda is NA help to estimate lambda. See details.

Details

lambda is calculated as a linear piecewise function which is equal to lambda_0 outside the [x_1,x_2] region. Inside this region, lambda is approximated by a line connecting points (x_1;lambda_1) and (x_2;lambda_2).

Value

An object of type data. Element

lambda	numeric vector containing an approximate mean magnitude of the signal.

is replaced with its new value.

set.SB *Set the coherent baseline*

Description

set.SB sets the baseline, describing coherent neutron scattering caused by uncorrelated atomic motion or any other baseline that needs to be preserved in the recovered signal.

Usage

```
set.SB(data, SB=NA, n.atoms=NA, scatter.length=NA, ADP=NA,
       fit=FALSE, oneADP=TRUE, ADP.lim = c(0, 0.05))
```

Arguments

data	an object of type data. See set.data for details.
SB	numeric vector which, if not NA, determines the baseline. See BBEST-package for details.
n.atoms, scatter.length, ADP	numerics. Specify the number of atoms of each atomtype in the unit cell, atomic scattering factors and atomic displacement parameters (ADP), respectively.
fit	logical, whether to fit ADP.
oneADP	logical. If TRUE a single parameter is used for all the APDs.
ADP.lim	numeric vector that specifies the lower and upper bounds for the fitted ADP.

Details

Baseline SB has to be specified. If no baseline is needed fill SB with zeroes. If n.atoms, scatter.length and ADP parameters are specified, the baseline is calculated according to

$$SB(x) = 1 - \frac{\sum_i N_i f_i^2 e^{-ADP_i x^2}}{N < f^2 >}(1 - \frac{< f >^2 - < f^2 >}{< f >^2}).$$

If ADP parameters are to be fitted, indicate n.atoms, scatter.length and set parameter fit to TRUE. Set oneADP to the desired value.

Value

An object of type data. Element

SB	numeric vector containing the baseline.

is replaced with its new value. Element

fitADP	a list of values.

might be added to describe the fit details.

---

set.sigma *Set the experimental uncertainty*

Description

This function either sets the pointwise experimental uncertainty or estimates it using aws library.

Usage

```
set.sigma(data, sigma=NA, x.bkg.only=NA, n.regions=10, hmax=250, sigma2=c(0.1))
```

Arguments

data	an object of type data. See set.data for details.
sigma	numeric vector which, if not NA, determines the pointwise experimental uncertainty.
x.bkg.only	if parameter sigma is NA, determines the peak-free region used to estimate the noise.
n.regions	if both parameters sigma and x.bkg.only are NA, the grid is split into n.regions equal regions. The noise is then estimated for each of these regions. See details
hmax	specifies the maximal bandwidth
sigma2	specifies the estimation of the signal's variance

Details

We assume the experimental uncertainty to have a Gaussian distribution with x-dependent amplitude. Splitting the grid into n.regions segments and estimating Gaussian standard deviations over each of these segments allows us to approximate the true distribution.

The function uses aws package that uses a Propagation-Separation Approach for signal smoothing. The use of sigma2 argument allows to obtain a smoother or rougher result.

Value

An object of type data. Elements

sigma	numeric vector containing the estimated noise level.
smoothed	if both parameters sigma and x.bkg.only are NA contains a smoothed estimate of the regression function.

are replaced with their new values.

References

Polzehl J, Papafitsoros K, Tabelow K (2020). Patch-Wise Adaptive Weights Smoothing in R. Journal of Statistical Software, 95(6), 1-27. Joerg Polzehl, Felix Anker (2020): aws: Adaptive Weights Smoothing. Version 2.5. https://CRAN.R-project.org/package=aws.

Examples

```
## Not run:
# Setting x and y
x <- seq(.7, 30, 0.01)
y <- sin(x)
# Adding x-dependent noise
y <- y + rnorm(sd=0.05+x/240, n=length(x))

# estimating noise
dat <- list(x=x, y=y)
dat <- set.sigma(dat, n.regions=1, sigma2 = 0.005)
# use
```

```
# dat <- set.sigma(dat, n.regions=5)
# to see the difference


# Plotting results: noisy function and a
# smoothed estimate +/- 2 standard deviations
plot(x, y, t="l")
lines(dat$x, dat$smoothed, col=3, lwd=2)
lines(dat$x, dat$smoothed+2*dat$sigma, col=2)
lines(dat$x, dat$smoothed-2*dat$sigma, col=2)
abline(v=seq(min(x), max(x),length=5), col=4)

## End(Not run)
```

`sqa.split` *Split .sqa file into individual files for each databank*

Description

`sqa.split` splits *PDFgetN .sqa*-file into individual files for each databank.

Usage

```
sqa.split(file = stop("'file' must be specified"))
```

Arguments

`file`	character, name of the source file.

See Also

`read.sqa`, `do.fit.banks`, `BBEST-package`

`test.signal` *A random function with a smooth background*

Description

`test.signal` creates a random function that consists of peaks, a smooth background, and a Gaussian noise.

Usage

```
test.signal(x, lambda, sigma, x.delta, knots.n, peaks.widthRange, peaks.n)
```

Arguments

x	numeric vector, the x-points where data should be generated.
lambda	numeric, the mean signal magnitude.
sigma	numeric, the noise level.
x.delta	numeric, the minimum spacing allowed between spline knots. Defines background smoothness.
knots.n	numeric, a number of spline knots to generate.
peaks.widthRange	numeric vector, specifies range in peak widths.
peaks.n	numeric, the number of peaks to generate.

Details

The background is calculated as a sum of fundamental splines on the randomly generated knots. The function is a sum of the background, random peaks, and Gaussian noise.

Value

An object of type data (see set.data) with the following elements added:

knots	list with elements x and y that specify the knot positions and knot values, respectively.
bkg	numeric vector containing the generated background.

Examples

```
# 1. Create test function
f <- test.signal(x=seq(0,30,0.01), lambda=5,
         sigma=0.1, x.delta=1.0, knots.n=5,
         peaks.widthRange=c(0.1, 0.3), peaks.n=7)

# 2. Plot results
plot(f$x, f$y, t="l", xlab="x", ylab="f")
lines(f$x, f$bkg, col=2)
lines(f$x, f$y - f$bkg, col="gray")
legend(20, .9*max(f$y), c("test function", "background",
       "peaks+noise"), lty=1, col=c(1,2,"gray"))
```

---

trim.data *Truncate data*

Description

The function truncates the data (deletes low- and high-x information).

Usage

```
trim.data(data, x.min, x.max)
```

Arguments

data	an object of type data. See set.data for details.
x.min, x.max	numeric values determining the region to keep.

Details

Frequently, the experimental data need to be truncated to remove unwanted ranges.

Value

an object of type data with all functions cropped to the region [x.min, x.max]

Examples

```
# prepare data
x <- seq(0, 50, 0.01)
y <- .8*exp(-x)*x^4
dat <- list(x=x, y=y)
# truncate
dat <- trim.data(dat, 1, 25)
# plot results
plot(x,y,t="l",lwd=4, col=4)
lines(dat$x, dat$y, lwd=4, col=2)
legend(15,3,c("initial", "truncated"), lty=1, col=c(4,2))
```

---

withProgress *Reporting progress (functional API)*

Description

Reports progress to the user during long-running operations.

Usage

```
withProgress(
  session,
  expr,
  min = 0,
  max = 1,
  env = parent.frame(),
  quoted = FALSE
)

setProgress(message = NULL, detail = NULL, value = NULL)
```

Arguments

session	The Shiny session object, as provided by shinyServer to the server function.
expr	The work to be done. This expression should contain calls to setProgress.
min	The value that represents the starting point of the progress bar. Must be less than max.
max	The value that represents the end of the progress bar. Must be greater than min.
env	The environment in which expr should be evaluated.
quoted	Whether expr is a quoted expression (this is not common).
message	A single-element character vector; the message to be displayed to the user, or NULL to hide the current message (if any).
detail	A single-element character vector; the detail message to be displayed to the user, or NULL to hide the current detail message (if any). The detail message will be shown with a de-emphasized appearance relative to message.
value	Single-element numeric vector; the value at which to set the progress bar, relative to min and max. NULL hides the progress bar, if it is currently visible.

Details

This package exposes two distinct programming APIs for working with progress. withProgress and setProgress together provide a simple function-based interface, while the Progress reference class provides an object-oriented API.

Use withProgress to wrap the scope of your work; doing so will cause a new progress panel to be created, and it will be displayed the first time setProgress is called. When withProgress exits, the corresponding progress panel will be removed.

Generally, withProgress/setProgress should be sufficient; the exception is if the work to be done is asynchronous (this is not common) or otherwise cannot be encapsulated by a single scope. In that case, you can use the Progress reference class.

See Also

progressInit, Progress

Examples

```
## Not run:
# server.R
shinyServer(function(input, output, session) {
  output$plot <- renderPlot({
    withProgress(session, min=1, max=15, {
      setProgress(message = 'Calculation in progress',
                  detail = 'This may take a while...')
      for (i in 1:15) {
        setProgress(value = i)
        Sys.sleep(0.5)
      }
    })
```

```
    plot(cars)
  })
})

## End(Not run)
```

`write.fit.results` *Save results of the fit*

Description

`write.fit.results` writes the returned value of `do.fit` to a specified text file.

Usage

```
write.fit.results(fit.results, file = stop("'file' must be specified"))
```

Arguments

`fit.results`	list, the return value of `do.fit`.
`file`	character, the filename for saving the data.

See Also

`do.fit`, `BBEST-package`

`write.fix` *Save a correction file for individual detector banks*

Description

`write.fix` writes corrections obtained using `do.fit.banks` to a specified file in a form suitable for *PDFgetN*.

Usage

```
write.fix(fit.results, file = stop("'file' must be specified"))
```

Arguments

`fit.results`	list, the return value of `do.fit.banks`.
`file`	character, the filename for saving the data.

See Also

`read.sqa`, `do.fit.banks`, `BBEST-package`

Index