Package 'Luminescence'

July 2, 2014

Type Package

Title Package for Luminescence Dating data analysis

Version 0.3.4

Date 2014-06-07

Author Sebastian Kreutzer [aut, trl, cre],Christoph Schmidt [aut],Margret C. Fuchs [aut],Michael Dietze [aut],Manfred Fischer [aut, trl],Christoph Burow [aut],Markus Fuchs [ths]

Maintainer Sebastian Kreutzer <sebastian.kreutzer@u-bordeaux-montaigne.fr>

Description
Package provides a collection of various R functions for Luminescence Dating data analysis.

Contact Package Developer Team <team@r-luminescence.de>

License GPL-3

Depends R (>= 3.1.0), utils, Rserve

Imports methods, XML, shape, rgl, matrixStats

URL http://CRAN.R-project.org/package=Luminescence

Collate Analyse_SAR.OSLdata.R analyse_SAR.CWOSL.R analyse_SAR.TL.R
analyse_IRSAR.RF.R CW2pLM.R CW2pLMi.R CW2pHMi.R CW2pPMi.R
calc_FadingCorr.R calc_FuchsLang2001.R calc_OSLLxTxRatio.R
calc_TLLxTxRatio.R Second2Gray.R fit_LMCurve.R fit_CWCurve.R
plot_Risoe.BINfileData.R plot_KDE.R plot_GrowthCurve.R
plot_Histogram.R plot_RadialPlot.R plot_RLum.R
plot_RLum.Analysis.R plot_RLum.Data.Curve.R readBIN2R.R
RisoeBINfileData-class.R Risoe.BINfileData2RLum.Analysis.R
RLum-class.R RLum.Data-class.R RLum.Data.Curve-class.R
RLum.Analysis-class.R RLum.Results-class.R calc_CentralDose.R
calc_FiniteMixture.R calc_MinDose3.R calc_MinDose4.R
calc_CommonDose.R calc_CosmicDoseRate.R merge_Risoe.BINfileData.R writeR2BIN.R
Risoe.BINfileData2RLum.Data.Curve.R calc_HomogeneityTest.R
calc_AliquotSize.R readXSYG2R.R RLum.Data.Spectrum-class.R

plot_RLum.Data.Spectrum.R calc_MaxDose3.R plot_AbanicoPlot.R
plot_DRTResults.R calc_Statistics.R apply_CosmicRayRemoval.R
apply_EfficiencyCorrection.R zzz.R

NeedsCompilation no

Repository CRAN

Date/Publication 2014-06-08 12:27:30

R topics documented:

`Luminescence-package` *Collection of functions for luminescence dating data analysis*

Description

This package provides various functions developed for the purpose of Luminescence Dating data analysis.

Details

Package:	Luminescence
Type:	Package
Version:	0.3.4
Date:	2014-06-07
License:	GPL-3

Author(s)

Authors

Christoph Burow	University of Cologne, Germany
Michael Dietze	GFZ Helmholtz Centre Potsdam, Germany
Manfred Fischer	University of Bayreuth, Germany
Margret C. Fuchs	Alfred Wegener Insitute for Polar and Marine Research, Potsdam, Germany
Sebastian Kreutzer	IRAMAT-CRP2A, Universite Bordeaux Montaigne, Pessac, France
Christoph Schmidt	University of Bayreuth, Germany

Supervisor

Markus Fuchs, Justus-Liebig-University Giessen, Germany

Support contact

http://forum.r-luminescence.de

<team@r-luminescence.de>

Bug reporting

<bugtracker@r-luminescence.de>

Project website

http://www.r-luminescence.de

Package maintainer

Sebastian Kreutzer, IRAMAT-CRP2A, Universite Bordeaux Montaigne, Pessac, France, <sebastian.kreutzer@u-bordeaux-montaigne.fr>

Acknowledgement

Cooperation and personal exchange between the developers is gratefully funded by the DFG (SCHM 3051/3-1) in the framework of the program "Scientific Networks". Project title: "Lum.Network: Ein Wissenschaftsnetzwerk zur Analyse von Lumineszenzdaten mit R" (2014-2016)

References

Dietze, M., Kreutzer, S., Fuchs, M.C., Burow, C., Fischer, M., Schmidt, C., 2013. A practical guide to the R package Luminescence. Ancient TL, 31, pp. 11-18.

Kreutzer, S., Schmidt, C., Fuchs, M.C., Dietze, M., Fischer, M., Fuchs, M., 2012. Introducing an R package for luminescence dating analysis. Ancient TL, 30, pp. 1-8.

`analyse_IRSAR.RF` *Analyse IRSAR RF measurements*

Description

Function to analyse IRSAR RF measurements on K-feldspar samples, performed using the protocol according to Erfurt et al. (2003)

Usage

```
analyse_IRSAR.RF(object, sequence.structure = c("NATURAL", "REGENERATED"),
    fit.range.min, fit.range.max, fit.trace = FALSE, fit.MC.runs = 10,
    output.plot = TRUE, xlab.unit = "s", legend.pos = "bottom",
    ...)
```

Arguments

`object`	`RLum.Analysis` (**required**): input object containing data for protocol analysis
`sequence.structure`	vector character (with default): specifies the general sequence structure. Allowed steps are `NATURAL`, `REGENERATED` In addition any other character is allowed in the sequence structure; such curves will be ignored.
`fit.range.min`	integer (optional): set the minimum channel range for signal fitting. Usually the entire data set is used for curve fitting, but there might be reasons to limit the channels used for fitting. Note: This option also limits the values used for natural signal calculation.
`fit.range.max`	integer (optional): set maximum channel range for signal fitting. Usually the entire data set is used for curve fitting, but there might be reasons to limit the channels used for fitting.
`fit.trace`	logical (with default): trace fitting (for debugging use)
`fit.MC.runs`	numeric (with default): set number of Monte Carlo runs for start parameter estimation. Note: Higher values will significantly increase the calculation time
`output.plot`	logical (with default): plot output (`TRUE` or `FALSE`)
`xlab.unit`	character (with default): set unit for x-axis
`legend.pos`	character (with default): useful keywords are `bottomright`, `bottom`, `bottomleft`, `left`, `topleft`, `top`, `topright`, `right` and `center`. For further details see `legend`.
`...`	further arguments that will be passed to the plot output. Currently supported arguments are `main`, `xlab`, `ylab`

Details

The function performs an IRSAR analysis described for feldspar samples by Erfurt et al. (2003) assuming a negligible sensitivity change of the RF signal.
General Sequence Structure (according to Erfurt et al. (2003))

1. Measuring IR-RF intensity of the natural dose for a few seconds ($D_{natural}$)
2. Bleach the samples under solar conditions for at least 30 min without changing the geometry
3. Waiting for at least one hour
4. Regeneration of the IR-RF signal to at least the natural level
5. Fitting data with a stretched exponential function
6. Calculate the the palaedose D using the parameters from the fitting

Function Used For The Fitting (according to Erfurt et al. (2003))

$$\phi(D) = \phi_0 - \Delta\phi(1 - exp(-\lambda * D))^{\beta}$$

with $\phi(D)$ the dose dependent IR-RF flux, ϕ_0 the inital IR-RF flux, $\Delta\phi$ the dose dependent change of the IR-RF flux, λ the exponential parameter, D the dose and β the dispersive factor.

To obtain the palaedose the function is changed to:

$$D = ln(-(\phi(D) - \phi_0)/(-\lambda * \phi)^{1/\beta} + 1)/ - \lambda$$

The fitting is done using the `port` algorithm of the `nls` function.

Value

A plot (optional) and an `RLum.Results` object is returned containing the following elements:

`De.values`	`data.frame` containing De-values with error (gray dashed lines in the plot) and further parameters
`fit`	`nls` `nlsModel` object

Note: The output (`De.values`) should be accessed using the function `get_RLum.Results`

Function version

0.1.3 (2014-04-13 14:21:09)

Note

This function assumes that there is no sensitivity change during the measurements (natural vs. regenerated signal), which is in contrast to the findings from Buylaert et al. (2012).

Author(s)

Sebastian Kreutzer, JLU Giessen/Freiberg Instruments (Germany)
R Luminescence Package Team

References

Buylaert, J.P., Jain, M., Murray, A.S., Thomsen, K.J., Lapp, T., 2012. IR-RF dating of sand-sized K-feldspar extracts: A test of accuracy. Radiation Measurements 1-7. doi: 10.1016/j.radmeas.2012.06.021

Erfurt, G., Krbetschek, M.R., 2003. IRSAR - A single-aliquot regenerative-dose dating protocol applied to the infrared radiofluorescence (IR-RF) of coarse- grain K-feldspar. Ancient TL 21, 35-42.

Erfurt, G., 2003. Infrared luminescence of Pb+ centres in potassium-rich feldspars. physica status solidi (a) 200, 429-438.

Erfurt, G., Krbetschek, M.R., 2003. Studies on the physics of the infrared radioluminescence of potassium feldspar and on the methodology of its application to sediment dating. Radiation Measurements 37, 505-510.

Erfurt, G., Krbetschek, M.R., Bortolot, V.J., Preusser, F., 2003. A fully automated multi-spectral radioluminescence reading system for geochronometry and dosimetry. Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 207, 487-499.

Trautmann, T., 2000. A study of radioluminescence kinetics of natural feldspar dosimeters: experiments and simulations. Journal of Physics D: Applied Physics 33, 2304-2310.

Trautmann, T., Krbetschek, M.R., Dietrich, A., Stolz, W., 1998. Investigations of feldspar radioluminescence: potential for a new dating technique. Radiation Measurements 29, 421-425.

Trautmann, T., Krbetschek, M.R., Dietrich, A., Stolz, W., 1999. Feldspar radioluminescence: a new dating method and its physical background. Journal of Luminescence 85, 45-58.

Trautmann, T., Krbetschek, M.R., Stolz, W., 2000. A systematic study of the radioluminescence properties of single feldspar grains. Radiation Measurements 32, 685-690.

See Also

RLum.Analysis, RLum.Results, get_RLum.Results, nls

Examples

```
##load data
data(ExampleData.RLum.Analysis, envir = environment())

##perform analysis
temp <- analyse_IRSAR.RF(object = IRSAR.RF.Data)
```

`analyse_SAR.CWOSL` *Analyse SAR CW-OSL measurements*

Description

The function performs a SAR CW-OSL analysis on a `RLum.Analysis` object including growth curve fitting.

Usage

```
analyse_SAR.CWOSL(object, signal.integral.min, signal.integral.max,
    background.integral.min, background.integral.max, rejection.criteria = list(
      recycling.ratio = 10,
      recuperation.rate = 10, palaeodose.error = 10), dose.points,
    log = "", output.plot = TRUE, output.plot.single = FALSE,
    ...)
```

Arguments

`object` `RLum.Analysis`(**required**): input object containing data for analysis

`signal.integral.min`
`integer` (**required**): lower bound of the signal integral

`signal.integral.max`
codeinteger (**required**): upper bound of the signal integral

`background.integral.min`
`integer` (**required**): lower bound of the background integral

`background.integral.max`
`integer` (**required**): upper bound of the background integral

`rejection.criteria`

`dose.points` numeric (optional): a numeric vector containg the dose points values Using this argument overwrites dose point values in the signal curves.

`log` character (with default): a character string which contains "x" if the x axis is to be logarithmic, "y" if the y-axis is to be logarithmic and "xy" or "yx" if both axes are to be logarithmic. See plot.default.

`output.plot` logical (with default): enables or disables plot output.

`output.plot.single`
logical (with default): single plot output (`TRUE/FALSE`) to allow for plotting the results in single plot windows. Requires `output.plot = TRUE`.

`...` further arguments that will be passed to the function `plot_GrowthCurve`

Details

The function performs an analysis for a standard SAR protocol measurements introduced by Murray and Wintle (2000) with CW-OSL curves. For the calculation of the Lx/Tx value the function calc_OSLLxTxRatio is used.

Provided rejection criteria

'recyling.ratio': calculated for every repeated regeneration dose point.
'recuperation.rate': recuperation rate calculated by comparing the Lx/Tx values of the zero regeneration point with the Ln/Tn value (the Lx/Tx ratio of the natural signal). For methodological background see Aitken and Smith (1988)

Value

A plot (optional) and an `RLum.Results` object is returned containing the following elements:

`De.values` data.frame containing De-values, De-error and further parameters

`LnLxTnTx.values` data.frame of all calculated Lx/Tx values including signal, background counts and the dose points.

`rejection.criteria` data.frame with values that might by used as rejection criteria. NA is produced if no R0 dose point exists.

The output should be accessed using the function `get_RLum.Results`

Function version

0.3.4 (2014-05-13 15:18:33)

Note

This function must not be mixed up with the function `Analyse_SAR.OSLdata`, which works with Risoe.BINfileData-class objects.

Author(s)

Sebastian Kreutzer, Freiberg Instruments/JLU Giessen (Germany)
R Luminescence Package Team

References

Aitken, M.J. & Smith, B.W., 1988. Optical dating: recuperation after bleaching. Quaternary Science Reviews, 7, pp. 387-393.

Duller, G., 2003. Distinguishing quartz and feldspar in single grain luminescence measurements. Radiation Measurements, 37 (2), pp. 161-165.

Murray, A.S. & Wintle, A.G., 2000. Luminescence dating of quartz using an improved single-aliquot regenerative-dose protocol. Radiation Measurements, 32, pp. 57-73.

See Also

calc_OSLLxTxRatio, plot_GrowthCurve, RLum.Analysis, RLum.Results get_RLum.Results

Examples

```
##load data
##ExampleData.BINfileData contains two BINfileData objects
##CWOSL.SAR.Data and TL.SAR.Data
data(ExampleData.BINfileData, envir = environment())

##transform the values from the first position in a RLum.Analysis object
object <- Risoe.BINfileData2RLum.Analysis(CWOSL.SAR.Data, pos=1)

##perform SAR analysis
analyse_SAR.CWOSL(object,
                  signal.integral.min = 1,
                  signal.integral.max = 2,
                  background.integral.min = 900,
                  background.integral.max = 1000,
                  log = "x",
                  fit.method = "EXP")
```

Analyse_SAR.OSLdata *Analyse SAR CW-OSL measurements.*

Description

The function analyses SAR CW-OSL curve data and provides a summary of the measured data for every position. The output of the function is optimised for SAR OSL measurements on quartz.

Usage

```
Analyse_SAR.OSLdata(input.data, signal.integral, background.integral,
    position, run, set, info.measurement = "unkown measurement",
    log = "", output.plot = FALSE, output.plot.single = FALSE,
    cex.global = 1)
```

Arguments

input.data	Risoe.BINfileData-class (**required**): input data from a Risoe BIN file, produced by the function readBIN2R.
signal.integral	vector (**required**): channels used for the signal integral, e.g. signal.integral=c(1:2)

`background.integral`
: vector (**required**): channels used for the background integral, e.g. `background.integral=c(85:100)`

`position`
: vector (optional): reader positions that want to be analysed (e.g. `position=c(1:48)`). Empty positions are automatically omitted. If no value is given all positions are analysed by default.

`run`
: vector (optional): range of runs used for the analysis. If no value is given the range of the runs in the sequence is deduced from the Risoe.BINfileData object.

`set`
: vector (optional): range of sets used for the analysis. If no value is given the range of the sets in the sequence is deduced from the `Risoe.BINfileData` object.

`info.measurement`
: character (with default): option to provide information about the measurement on the plot output (e.g. name of the BIN or BINX file).

`log`
: character (with default): a character string which contains "x" if the x axis is to be logarithmic, "y" if the y axis is to be logarithmic and "xy" or "yx" if both axes are to be logarithmic. See plot.default.

`output.plot`
: logical (with default): plot output (`TRUE/FALSE`)

`output.plot.single`
: logical (with default): single plot output (`TRUE/FALSE`) to allow for plotting the results in single plot windows. Requires `output.plot = TRUE`.

`cex.global`
: numeric (with default): global scaling factor.

Details

The function works only for standard SAR protocol measurements introduced by Murray and Wintle (2000) with CW-OSL curves. For the calculation of the Lx/Tx value the function calc_OSLLxTxRatio is used.

Provided rejection criteria

'recyling ratio': calculated for every repeated regeneration dose point.
'recuperation': recuperation rate calculated by comparing the Lx/Tx values of the zero regeneration point with the Ln/Tn value (the Lx/Tx ratio of the natural signal). For methodological background see Aitken and Smith (1988)

'IRSL/BOSL': the integrated counts (`signal.integral`) of an IRSL curve are compared to the integrated counts of the first regenerated dose point. It is assumed that IRSL curves got the same dose as the first regenerated dose point. **Note:** This is not the IR depletation ratio described by Duller (2003).

Value

A plot (optional) and list is returned containing the following elements:

`LnLxTnTx`
: data.frame of all calculated Lx/Tx values including signal, background counts and the dose points.

`RejectionCriteria`
: data.frame with values that might by used as rejection criteria. NA is produced if no R0 dose point exists.

`SARParameters`
: data.frame of additional measurement parameters obtained from the BIN file, e.g. preheat or read temperature (not valid for all types of measurements).

Function version

0.2.12 (2014-04-13 14:22:24)

Note

Rejection criteria are calculated but not considered during the analysis to discard values.

The development of this function will not be continued. We recommend to use the function analyse_SAR.CWOSL instead.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany), Margret C. Fuchs, AWI Potsdam (Germany), R Luminescence Package Team

References

Aitken, M.J. & Smith, B.W., 1988. Optical dating: recuperation after bleaching. Quaternary Science Reviews, 7, pp. 387-393.

Duller, G., 2003. Distinguishing quartz and feldspar in single grain luminescence measurements. Radiation Measurements, 37 (2), pp. 161-165.

Murray, A.S. & Wintle, A.G., 2000. Luminescence dating of quartz using an improved single-aliquot regenerative-dose protocol. Radiation Measurements, 32, pp. 57-73.

See Also

calc_OSLLxTxRatio, Risoe.BINfileData-class, readBIN2R

and for further analysis plot_GrowthCurve

Examples

```
##load data
data(ExampleData.BINfileData, envir = environment())

##analyse data
output <- Analyse_SAR.OSLdata(input.data = CWOSL.SAR.Data,
                              signal.integral = c(1:5),
                              background.integral = c(900:1000),
                              position = c(1:1),
                              output.plot = TRUE)
```

```
##combine results relevant for further analysis
output.SAR <- data.frame(Dose = output$LnLxTnTx[[1]]$Dose,
                         LxTx = output$LnLxTnTx[[1]]$LxTx,
                         LxTx.Error = output$LnLxTnTx[[1]]$LxTx.Error)
output.SAR
```

`analyse_SAR.TL` *Analyse SAR TL measurements*

Description

The function performs an SAR TL analysis on a `RLum.Analysis` object including growth curve fitting.

Usage

```
analyse_SAR.TL(object, object.background, signal.integral.min,
    signal.integral.max, sequence.structure = c("PREHEAT", "SIGNAL",
        "BACKGROUND"), rejection.criteria = list(recycling.ratio = 10,
        recuperation.rate = 10), log = "", ...)
```

Arguments

`object`	`RLum.Analysis`(**required**): input object containing data for analysis
`object.background`	currently not used
`signal.integral.min`	integer (**required**): requires the channel number for the lower signal integral bound (e.g. `signal.integral.min = 100`)
`signal.integral.max`	integer (**required**): requires the channel number for the upper signal integral bound (e.g. `signal.integral.max = 200`)
`sequence.structure`	vector character (with default): specifies the general sequence structure. Three steps are allowed `"PREHEAT"`, `"SIGNAL"`, `"BACKGROUND"`, in addition a parameter `"EXCLUDE"`. This allows excluding TL curves which are not relevant for the protocol analysis. (Note: None TL are removed by default)
`rejection.criteria`	list (with default): list containing rejection criteria in percentage for the calculation.
`log`	character (with default): a character string which contains "x" if the x axis is to be logarithmic, "y" if the y axis is to be logarithmic and "xy" or "yx" if both axes are to be logarithmic. See plot.default).
`...`	further arguments that will be passed to the function `plot_GrowthCurve`

Details

This function performs a SAR TL analysis on a set of curves. The SAR procedure in general is given by Murray and Wintle (2000). For the calculation of the Lx/Tx value the function calc_TLLxTxRatio is used.

Provided rejection criteria

'recyling.ratio': calculated for every repeated regeneration dose point.
'recuperation.rate': recuperation rate calculated by comparing the Lx/Tx values of the zero regeneration point with the Ln/Tn value (the Lx/Tx ratio of the natural signal). For methodological background see Aitken and Smith (1988)

Value

A plot (optional) and an `RLum.Results` object is returned containing the following elements:

`De.values` data.frame containing De-values and further parameters

`LnLxTnTx.values` data.frame of all calculated Lx/Tx values including signal, background counts and the dose points.

`rejection.criteria` data.frame with values that might by used as rejection criteria. NA is produced if no R0 dose point exists.

note: the output should be accessed using the function `get_RLum.Results`

Function version

0.1.4 (2014-04-14 02:17:38)

Note

THIS IS A BETA VERSION

None TL curves will be removed from the input object without further warning.

Author(s)

Sebastian Kreutzer, Freiberg Instruments/JLU Giessen (Germany)
R Luminescence Package Team

References

Aitken, M.J. & Smith, B.W., 1988. Optical dating: recuperation after bleaching. Quaternary Science Reviews, 7, pp. 387-393.

Murray, A.S. & Wintle, A.G., 2000. Luminescence dating of quartz using an improved single-aliquot regenerative-dose protocol. Radiation Measurements, 32, pp. 57-73.

See Also

calc_TLLxTxRatio, plot_GrowthCurve, RLum.Analysis, RLum.Results get_RLum.Results

Examples

```
##load data
data(ExampleData.BINfileData, envir = environment())

##transform the values from the first position in a RLum.Analysis object
object <- Risoe.BINfileData2RLum.Analysis(TL.SAR.Data, pos=3)

##perform analysis
analyse_SAR.TL(object,
               signal.integral.min = 210,
               signal.integral.max = 220,
               log = "y",
               fit.method = "EXP OR LIN",
               sequence.structure = c("SIGNAL", "BACKGROUND"))
```

---

apply_CosmicRayRemoval

Function to remove cosmic rays from an RLum.Data.Spectrum S4 class objects

---

Description

The function provides several methods for cosmic ray removal and spectrum smoothing for an RLum.Data.Spectrum S4 class objects

Usage

```
apply_CosmicRayRemoval(object, method = "Pych", method.Pych.smoothing = 2,
    method.Pych.histogram.plot = FALSE, silent = FALSE, ...)
```

Arguments

object	RLum.Data.Spectrum (**required**): S4 object of class RLum.Data.Spectrum
method	character (with default): Defines method that is applied for cosmic ray removal. Allowed methods smooth (smooth), smooth.spline (smooth.spline) and Pych (default). See details for further information.
method.Pych.smoothing	integer (with default): Smoothing parameter for cosmic ray removal according to Pych (2003). The value defines how many neighboring values in each frame are used for smoothing (e.g. 2 means that the two previous and two following values are used)

`method.Pych.histogram.plot`
: logical (with default): If TRUE the histograms used for the cosmic-ray removal are returned as plot including the used threshold. Note: A separat plot is returned for each frame!,

`silent`
: logical (with default): Option to suppress terminal output

`...`
: further arguments and graphical parameters that will be passed to the smooth function.

Details

`method = "Pych"`

This method applies the cosmic-ray removal algorithm described by Pych (2003). Some aspects that are different to the publication:

- For interpolation between neighbouring values the median and not the mean is used.
- The number of breaks to construct the histogram is set to: `length(number.of.input.values)/2`

For further details see references below.

`method = "smooth"`

Method uses the function smooth to remove cosmic rays.

Arguments that can be passed are: `kind`, `twiceit`

`method = "smooth.spline"`
Method uses the function `smooth.spline` to remove cosmic rays.
Arguments that can be passed are: `spar`

How to combine methods?

Different methods can combined by applying the method repeatedly on the dataset (see example).

Value

Returns same object as input (`RLum.Data.Spectrum`)

Function version

0.1.2 (2014-06-04 18:50:49)

Note

This function has BETA status

Author(s)

Sebastian Kreutzer, Universite Bordeaux Montaigne (France) R Luminescence Package Team

References

Pych, W., 2003. A Fast Algorithm for Cosmic-Ray Removal from Single Images. Astrophysics 116, 148-153. http://arxiv.org/pdf/astro-ph/0311290.pdf?origin=publication_detail

See Also

RLum.Data.Spectrum, smooth, smooth.spline, apply_CosmicRayRemoval

Examples

```
##(1) - use with your own data and combine (uncomment for usage)
## run two times the default method and smooth with another method
## your.spectrum <- apply_CosmicRayRemoval(your.spectrum, method = "Pych")
## your.spectrum <- apply_CosmicRayRemoval(your.spectrum, method = "Pych")
## your.spectrum <- apply_CosmicRayRemoval(your.spectrum, method = "smooth")
```

---

apply_EfficiencyCorrection

Function to apply spectral efficiency correction on RLum.Data.Spectrum S4 class objects

---

Description

The function allows spectral efficiency corrections for RLum.Data.Spectrum S4 class objects

Usage

```
apply_EfficiencyCorrection(object, spectral.efficiency)
```

Arguments

object	RLum.Data.Spectrum (**required**): S4 object of class RLum.Data.Spectrum
spectral.efficiency	data.frame (**required**): Data set containg wavelengths (x-column) and relative spectral response values (y-column) in percentage

Details

The efficiency correction is based on a spetral response dataset provided by the user. Usually the data set for the quantum efficiency is of lower resolution and values are interpolated for the required spectral resolution.

Value

Returns same object as input (RLum.Data.Spectrum)

Function version

0.1 (2014-04-13 14:27:05)

Note

Please note that the spectral efficiency data from the camera may not sufficiently correct for spectral efficiency of the entire optical system (e.g., spectrometer, camera ...).

This function has BETA status

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany), Johannes Friedrich, University of Bayreuth (Germany) R Luminescence Package Team

References

-

See Also

RLum.Data.Spectrum

Examples

```
##(1) - use with your own data (uncomment for usage)
## spectral.efficiency <- read.csv("your data")
##
## your.spectrum <- apply_EfficiencyCorrection(your.spectrum, )
```

BaseDataSet.CosmicDoseRate

Base data set for cosmic dose rate calculation

Description

Collection of data from various sources needed for cosmic dose rate calculation

Usage

```
BaseDataSet.CosmicDoseRate
```

Format

values.cosmic.Softcomp:	data frame containing cosmic dose rates for shallow depths (< 167 g cm^-2) obtained using the
values.factor.Altitude:	data frame containing altitude factors for adjusting geomagnetic field-change factors. Values w
values.par.FJH:	data frame containing values for parameters F, J and H (read from Fig. 2 in Prescott & Hutton 1

$$Dc = D0 * (F + J * exp((altitude/1000)/H))$$

Version

0.1

Source

The following data were carefully read from figures in mentioned sources and used for fitting procedures. The derived expressions are used in the function `calc_CosmicDoseRate`.

values.cosmic.Softcomp

Program:	"AGE"
Reference:	Gruen (2009)
Fit:	Polynomials in the form of

For depths between 40-167 g cm^-2:

$$y = 2 * 10^-6 * x^2 - 0.0008 * x + 0.2535$$

(For depths <40 g cm^-2)

$$y = -6 * 10^-8 * x^3 + 2 * 10^-5 * x^2 - 0.0025 * x + 0.2969$$

values.factor.Altitude

Reference:	Prescott & Hutton (1994)
Page:	499
Figure:	1
Fit:	2-degree polynomial in the form of

$$y = -0.026 * x^2 + 0.6628 * x + 1.0435$$

values.par.FJH

Reference:	Prescott & Hutton (1994)
Page:	500
Figure:	2
Fits:	3-degree polynomials and linear fits

F (non-linear part, λ < 36.5 deg.):

$$y = -7 * 10^-7 * x^3 - 8 * 10^-5 * x^2 - 0.0009 * x + 0.3988$$

F (linear part, λ > 36.5 deg.):

$$y = -0.0001 * x + 0.2347$$

J (non-linear part, λ < 34 deg.):

$$y = 5 * 10^-6 * x^3 - 5 * 10^-5 * x^2 + 0.0026 * x + 0.5177$$

J (linear part, λ > 34 deg.):

$$y = 0.0005 * x + 0.7388$$

H (non-linear part, λ < 36 deg.):

$$y = -3 * 10^-6 * x^3 - 5 * 10^-5 * x^2 - 0.0031 * x + 4.398$$

H (linear part, λ > 36 deg.):

$$y = 0.0002 * x + 4.0914$$

References

Gruen, R., 2009. The "AGE" program for the calculation of luminescence age estimates. Ancient TL, 27, pp. 45-46.

Prescott, J.R., Hutton, J.T., 1988. Cosmic ray and gamma ray dosimetry for TL and ESR. Nuclear Tracks and Radiation Measurements, 14, pp. 223-227.

Prescott, J.R., Hutton, J.T., 1994. Cosmic ray contributions to dose rates for luminescence and ESR dating: large depths and long-term time variations. Radiation Measurements, 23, pp. 497-500.

Examples

```
##load data
data(BaseDataSet.CosmicDoseRate)
```

`calc_AliquotSize` *Estimate the amount of grains on an aliquot*

Description

Estimate the number of grains on an aliquot. Alternatively, the packing density of an aliquot is computed.

Usage

```
calc_AliquotSize(grain.size, sample.diameter, packing.density = 0.65,
    MC.estimate = TRUE, grains.counted, ...)
```

Arguments

`grain.size`	`numeric` (**required**): mean grain size (microns) or a range of grain sizes from which the mean grain size is computed (e.g. `c(100,200)`).
`sample.diameter`	`numeric` (**required**): diameter (mm) of the targeted area on the sample carrier.
`packing.density`	`numeric` (with default) empirical value for mean packing density. If `packing.density = "inf"` a hexagonal structure on an infinite plane with a packing density of $0.906\ldots$ is assumed.
`MC.estimate`	`logical` (optional): if `TRUE` the function performs a monte carlo simulation for estimating the amount of grains on the sample carrier and assumes random errors in grain size distribution and packing density. Requires a vector with min and max grain size for `grain.size`. For more information see details.
`grains.counted`	`numeric` (optional) grains counted on a sample carrier. If a non-zero positive integer is provided this function will calculate the packing density of the aliquot. If more than one value is provided the mean packing density and its standard deviation is calculated. Note that this overrides `packing.density`.
`...`	further arguments to pass (`main, xlab, MC.iter`).

Details

This function can be used to either estimate the number of grains on an aliquot or to compute the packing density depending on the the arguments provided.
The following function is used to estimate the number of grains `n`:

$$n = (\pi * x^2)/(\pi * y^2) * d$$

where `x` is the radius of the aliquot size (microns), `y` is the mean radius of the mineral grains (mm) and `d` is the packing density (value between 0 and 1).

Packing density

The default value for `packing.density` is 0.65, which is the mean of empirical values determined by Heer et al. (2012) and unpublished data from the Cologne luminescence laboratory. If `packing.density = "inf"` a maximum density of $\pi/\sqrt{12} = 0.9068\ldots$ is used. However, note that this value is not appropriate as the standard preparation procedure of aliquots resembles a PECC ("Packing Equal Circles in a Circle") problem where the maximum packing density is asymptotic to about 0.87.

Monte Carlo simulation

The number of grains on an aliquot can be estimated by Monte Carlo simulation when setting `MC.estimate = TRUE`. Each of the parameters necessary to calculate `n` (`x`, `y`, `d`) are assumed to be normally distributed with means μ_x, μ_y, μ_d and standard deviations $\sigma_x, \sigma_y, \sigma_d$.

For the mean grain size random samples are taken first from $N(\mu_y, \sigma_y)$, where $\mu_y = mean.grain.size$ and $\sigma_y = (max.grain.size - min.grain.size)/4$ so that 95% of all grains are within the provided

the grain size range. This effectively takes into account that after sieving the sample there is still a small chance of having grains smaller or larger than the used mesh sizes. For each random sample the mean grain size is calculated, from which random subsamples are drawn for the Monte Carlo simulation.

The packing density is assumed to be normally distributed with an empirically determined $\mu = 0.65$ (or provided value) and $\sigma = 0.18$. The normal distribution is truncated at `d = 0.87` as this is approximately the maximum packing density that can be achieved in PECC problem.

The sample diameter has $\mu = sample.diameter$ and $\sigma = 0.2$ to take into account variations in sample disc preparation (i.e. applying silicon spray to the disc). A lower truncation point at `x = 0.5` is used, which assumes that aliqouts with smaller sample diameters of 0.5 mm are discarded. Likewise, the normal distribution is truncated at 9.8 mm, which is the diameter of the sample disc.

For each random sample drawn from the normal distributions the amount of grains on the aliquot is calculated. By default, `10^5` iterations are used, but can be reduced/increased with `MC.iter` (see `...`). The results are visualised in a bar- and boxplot together with a statistical summary.

Value

Returns terminal output. In addition an `RLum.Results` object is returned containing the following element:

`results` data.frame with calculation results.

The output should be accessed using the function `get_RLum.Results`

Function version

0.3 (2014-04-13 14:27:11)

Author(s)

Christoph Burow, University of Cologne (Germany)
R Luminescence Package Team

References

Duller, G.A.T., 2008. Single-grain optical dating of Quaternary sediments: why aliquot size matters in luminescence dating. Boreas 37, pp. 589-612.

Heer, A.J., Adamiec, G., Moska, P., 2012. How many grains are there on a single aliquot?. Ancient TL, 30, pp. 9-16.

Further reading

Chang, H.-C., Wang, L.-C., 2010. A simple proof of Thue's Theorem on Circle Packing. http://arxiv.org/pdf/1009.4322v1.pdf, 2013-09-13.

Graham, R.L., Lubachevsky, B.D., Nurmela, K.J., Oestergard, P.R.J., 1998. Dense packings of

congruent circles in a circle. Discrete Mathematics, 181, pp. 139-154.

Huang, W., Ye, T., 2011. Global optimization method for finding dense packings of equal circles in a circle. European Journal of Operational Research, 210, pp. 474-481.

Examples

```
## Estimate the amount of grains on a small aliquot
calc_AliquotSize(grain.size = c(100,150), sample.diameter = 1)

## Calculate the mean packing density of large aliquots
calc_AliquotSize(grain.size = c(100,200), sample.diameter = 8,
                 grains.counted = c(2525,2312,2880))
```

---

calc_CentralDose *Apply the central age model (CAM) after Galbraith et al. (1999) to a given De distribution*

Description

This function calculates the central dose and dispersion of the De distribution, their standard errors and the profile log likelihood function for sigma.

Usage

```
calc_CentralDose(input.data, sigmab = 0, sample.id = "unknown sample",
    print.iterations = FALSE, output.plot = TRUE)
```

Arguments

input.data	RLum.Results or data.frame (**required**): for data.frame: two columns with De (input.data[,1]) and De error (values[,2])
sigmab	numeric (with default): spread in De values given as a fraction (e.g. 0.2). This value represents the expected overdispersion in the data should the sample be well-bleached (Cunningham & Walling 2012, p. 100).
sample.id	character (with default): sample id
print.iterations	logical (with default): terminal output of calculation iterations
output.plot	logical (with default): plot output

Details

This function uses the equations of Galbraith et al. (1999, pp. 358-359). The parameter sigma is estimated using the maximum likelihood approach. A detailed explanation on maximum likelihood estimation can be found in the appendix of Galbraith & Laslett (1993, pp. 468-470)

Value

Returns a plot (optional) and terminal output. In addition an RLum.Results object is returned containing the following element:

`results` data.frame with statistical parameters.

The output should be accessed using the function get_RLum.Results

Function version

1.23[2014-04-29] (2014-04-29 18:44:00)

Author(s)

Christoph Burow, University of Cologne (Germany)
Based on a rewritten S script of Rex Galbraith, 2010

R Luminescence Package Team

References

Galbraith, R.F. & Laslett, G.M., 1993. Statistical models for mixed fission track ages. Nuclear Tracks Radiation Measurements, 4, pp. 459-470.

Galbraith, R.F., Roberts, R.G., Laslett, G.M., Yoshida, H. & Olley, J.M., 1999. Optical dating of single grains of quartz from Jinmium rock shelter, northern Australia. Part I: experimental design and statistical models. Archaeometry, 41, pp. 339-364.

Galbraith, R.F. & Roberts, R.G., 2012. Statistical aspects of equivalent dose and error calculation and display in OSL dating: An overview and some recommendations. Quaternary Geochronology, 11, pp. 1-27.

Further reading

Arnold, L.J. & Roberts, R.G., 2009. Stochastic modelling of multi-grain equivalent dose (De) distributions: Implications for OSL dating of sediment mixtures. Quaternary Geochronology, 4, pp. 204-230.

Bailey, R.M. & Arnold, L.J., 2006. Statistical modelling of single grain quartz De distributions and an assessment of procedures for estimating burial dose. Quaternary Science Reviews, 25, pp. 2475-2502.

Cunningham, A.C. & Wallinga, J., 2012. Realizing the potential of fluvial archives using robust OSL chronologies. Quaternary Geochronology, 12, pp. 98-106.

Rodnight, H., Duller, G.A.T., Wintle, A.G. & Tooth, S., 2006. Assessing the reproducibility and accuracy of optical dating of fluvial deposits. Quaternary Geochronology, 1, pp. 109-120.

Rodnight, H., 2008. How many equivalent dose values are needed to obtain a reproducible distribution?. Ancient TL, 26, pp. 3-10.

See Also

plot, calc_CommonDose, calc_FiniteMixture, calc_FuchsLang2001, calc_MinDose3, calc_MinDose4

Examples

```
##load example data
data(ExampleData.DeValues, envir = environment())

##apply the central dose model
calc_CentralDose(ExampleData.DeValues)
```

---

calc_CommonDose *Apply the (un-)logged common age model after Galbraith et al. (1999) to a given De distribution*

Description

Function to calculate the common dose of a De distribution.

Usage

```
calc_CommonDose(input.data, sigmab = 0, log = TRUE, sample.id = "unknown sample")
```

Arguments

input.data	RLum.Results or data.frame (**required**): for data.frame: two columns with De (input.data[,1]) and De error (values[,2])
sigmab	numeric (with default): spread in De values given as a fraction (e.g. 0.2). This value represents the expected overdispersion in the data should the sample be well-bleached (Cunningham & Walling 2012, p. 100).
log	logical (with default): fit the (un-)logged common age model to De data
sample.id	character (with default): sample id

Details

(Un-)logged model

When log = TRUE this function calculates the weighted mean of logarithmic De values. Each of the estimates is weighted by the inverse square of its relative standard error. The weighted mean is then transformed back to the dose scale (Galbraith & Roberts 2012, p. 14).

The log transformation is not applicable if the De estimates are close to zero or negative. In this case

the un-logged model can be applied instead (`log = FALSE`). The weighted mean is then calculated using the un-logged estimates of De and their absolute standard error (Galbraith & Roberts 2012, p. 14).

Value

Returns a terminal output. In addition an `RLum.Results` object is returned containing the following element:

`results` data.frame with statistical parameters.

The output should be accessed using the function `get_RLum.Results`

Function version

1.2 (2014-04-13 14:27:25)

Author(s)

Christoph Burow, University of Cologne (Germany)
R Luminescence Package Team

References

Galbraith, R.F. & Laslett, G.M., 1993. Statistical models for mixed fission track ages. Nuclear Tracks Radiation Measurements, 4, pp. 459-470.

Galbraith, R.F., Roberts, R.G., Laslett, G.M., Yoshida, H. & Olley, J.M., 1999. Optical dating of single grains of quartz from Jinmium rock shelter, northern Australia. Part I: experimental design and statistical models. Archaeometry, 41, pp. 339-364.

Galbraith, R.F. & Roberts, R.G., 2012. Statistical aspects of equivalent dose and error calculation and display in OSL dating: An overview and some recommendations. Quaternary Geochronology, 11, pp. 1-27.

Further reading

Arnold, L.J. & Roberts, R.G., 2009. Stochastic modelling of multi-grain equivalent dose (De) distributions: Implications for OSL dating of sediment mixtures. Quaternary Geochronology, 4, pp. 204-230.

Bailey, R.M. & Arnold, L.J., 2006. Statistical modelling of single grain quartz De distributions and an assessment of procedures for estimating burial dose. Quaternary Science Reviews, 25, pp. 2475-2502.

Cunningham, A.C. & Wallinga, J., 2012. Realizing the potential of fluvial archives using robust OSL chronologies. Quaternary Geochronology, 12, pp. 98-106.

Rodnight, H., Duller, G.A.T., Wintle, A.G. & Tooth, S., 2006. Assessing the reproducibility and accuracy of optical dating of fluvial deposits. Quaternary Geochronology, 1, pp. 109-120.

Rodnight, H., 2008. How many equivalent dose values are needed to obtain a reproducible distribution?. Ancient TL, 26, pp. 3-10.

See Also

calc_CentralDose, calc_FiniteMixture, calc_FuchsLang2001, calc_MinDose3, calc_MinDose4

Examples

```
## load example data
data(ExampleData.DeValues, envir = environment())

## apply the common dose model
calc_CommonDose(ExampleData.DeValues)
```

---

calc_CosmicDoseRate *Calculate the cosmic dose rate*

Description

This function calculates the cosmic dose rate taking into account the soft- and hard-component of the cosmic ray flux and allows corrections for geomagnetic latitude, altitude above sea-level and geomagnetic field changes.

Usage

```
calc_CosmicDoseRate(depth, density, latitude, longitude, altitude,
    corr.fieldChanges = FALSE, est.age = NA, half.depth = FALSE,
    error = 10)
```

Arguments

depth	numeric (**required**): depth of overburden (m). For more than one absorber use c(depth_1, depth_2, ..., depth_n)
density	numeric (**required**): average overburden density (g/cm^3). For more than one absorber use c(density_1, density_2, ..., density_n)
latitude	numeric (**required**): latitude (decimal degree), N positive
longitude	numeric (**required**): longitude (decimal degree), E positive
altitude	numeric (**required**): altitude (m above sea-level)
corr.fieldChanges	logical (with default): correct for geomagnetic field changes after Prescott & Hutton (1994). Apply only when justified by the data.
est.age	numeric (with default): estimated age range (ka) for geomagnetic field change correction (0-80 ka allowed)

`half.depth`	`logical` (with default): How to overcome with varying overburden thickness. If `TRUE` only half the depth is used for calculation. Apply only when justified, i.e. when a constant sedimentation rate can safely be assumed.
`error`	`numeric` (with default): general error (percentage) to be implemented on corrected cosmic dose rate estimate

Details

This function calculates the total cosmic dose rate considering both the soft- and hard-component of the cosmic ray flux.

Internal calculation steps

(1) Calculate total depth of all absorber in hg/cm^2 (1 hg/cm^2 = 100 g/cm^2)

$$absorber = depth_1 * density_1 + depth_2 * density_2 + ... + depth_n * density_n$$

(2) If `half.depth = TRUE`

$$absorber = absorber/2$$

(3) Calculate cosmic dose rate at sea-level and 55 deg. latitude

a) If absorber is > 167 g/cm^2 (only hard-component; Allkofer et al. 1975): apply equation given by Prescott & Hutton (1994) (c.f. Barbouti & Rastin 1983)

$$D0 = C/(((absorber + d)^{\alpha} + a) * (absober + H)) * exp(-B * absorber)$$

b) If absorber is < 167 g/cm^2 (soft- and hard-component): derive D0 from Fig. 1 in Prescott & Hutton (1988).

(4) Calculate geomagnetic latitude (Prescott & Stephan 1982, Prescott & Hutton 1994)

$$\lambda = arcsin(0.203 * cos(latitude) * cos(longitude - 291) + 0.979 * sin(latitude))$$

(5) Apply correction for geomagnetic latitude and altitude above sea-level. Values for F, J and H were read from Fig. 3 shown in Prescott & Stephan (1982) and fitted with 3-degree polynomials for lambda < 35 degree and a linear fit for lambda > 35 degree.

$$Dc = D0 * (F + J * exp((altitude/1000)/H))$$

(6) Optional: Apply correction for geomagnetic field changes in the last 0-80 ka (Prescott & Hutton 1994). Correction and altitude factors are given in Table 1 and Fig. 1 in Prescott & Hutton (1994). Values for altitude factor were fitted with a 2-degree polynomial. The altitude factor is operated on the decimal part of the correction factor.

$$Dc' = Dc * correctionFactor$$

Usage of `depth` **and** `density`

(1) If only one value for depth and density is provided, the cosmic dose rate is calculated for exactly one sample and one absorber as overburden (i.e. `depth*density`).

(2) In some cases it might be useful to calculate the cosmic dose rate for a sample that is overlain by more than one absorber, e.g. in a profile with soil layers of different thickness and a distinct difference in density. This can be calculated by providing a matching number of values for `depth` and `density` (e.g. `depth = c(1, 2),density = c(1.7, 2.4)`)

(3) Another possibility is to calculate the cosmic dose rate for more than one sample of the same profile. This is done by providing more than one values for `depth` and only one for `density`. For example, `depth = c(1, 2, 3), density = 1.7` will calculate the cosmic dose rate for three samples in 1, 2 and 3 m depth in a sediment of density 1.7 g/cm^3.

Value

Returns terminal output. In addition an `RLum.Results` object is returned containing the following element:

`results` data.frame with statistical parameters.

The output should be accessed using the function `get_RLum.Results`

Function version

0.5.2 (2014-04-13 14:27:31)

Note

Despite its universal use the equation to calculate the cosmic dose rate provided by Prescott & Hutton (1994) is falsely stated to be valid from the surface to 10^4 hg/cm^2 of standard rock. The original expression by Barbouti & Rastin (1983) only considers the muon flux (i.e. hard-component) and is by their own definition only valid for depths between 10-10^4 hg/cm^2.

Thus, for near-surface samples (i.e. for depths < 167 g/cm^2) the equation of Prescott & Hutton (1994) underestimates the total cosmic dose rate, as it neglects the influence of the soft-component of the cosmic ray flux. For samples at zero depth and at sea-level the underestimation can be as large as ~0.1 Gy/ka. In a previous article, Prescott & Hutton (1988) give another approximation of Barbouti & Rastins equation in the form of

$$D = 0.21 * exp(-0.070 * absorber + 0.0005 * absorber^2)$$

which is valid for depths between 150-5000 g/cm^2. For shallower depths (< 150 g/cm^2) they provided a graph (Fig. 1) from which the dose rate can be read.

As a result, this function employs the equation of Prescott & Hutton (1994) only for depths > 167 g/cm^2, i.e. only for the hard-component of the cosmic ray flux. Cosmic dose rate values for depths < 167 g/cm^2 were obtained from the "AGE" programm (Gruen 2009) and fitted with a 6-degree polynomial curve (and hence reproduces the graph shown in Prescott & Hutton 1988). However, these values assume an average overburden density of 2 g/cm^3.

It is currently not possible to obtain more precise cosmic dose rate values for near-surface samples as there is no equation known to the author of this function at the time of writing.

Author(s)

Christoph Burow, University of Cologne (Germany)
R Luminescence Package Team

References

Allkofer, O.C., Carstensen, K., Dau, W.D., Jokisch, H., 1975. Letter to the editor. The absolute cosmic ray flux at sea level. Journal of Physics G: Nuclear and Particle Physics, 1, pp. L51-L52.

Barbouti, A.I., Rastin, B.C., 1983. A study of the absolute intensity of muons at sea level and under various thicknesses of absorber. Journal of Physics G: Nuclear and Particle Physics, 9, pp. 1577-1595.

Crookes, J.N., Rastin, B.C., 1972. An investigation of the absolute intensity of muons at sea-level. Nuclear Physics B, 39, pp. 493-508.

Gruen, R., 2009. The "AGE" program for the calculation of luminescence age estimates. Ancient TL, 27, pp. 45-46.

Prescott, J.R., Hutton, J.T., 1988. Cosmic ray and gamma ray dosimetry for TL and ESR. Nuclear Tracks and Radiation Measurements, 14, pp.

223-227. Prescott, J.R., Hutton, J.T., 1994. Cosmic ray contributions to dose rates for luminescence and ESR dating: large depths and long-term time variations. Radiation Measurements, 23, pp. 497-500.

Prescott, J.R., Stephan, L.G., 1982. The contribution of cosmic radiation to the environmental dose for thermoluminescence dating. Latitude, altitude and depth dependences. PACT, 6, pp. 17-25.

See Also

BaseDataSet.CosmicDoseRate

Examples

```
##(1) calculate cosmic dose rate (one absorber)
calc_CosmicDoseRate(depth = 2.78, density = 1.7,
                    latitude = 38.06451, longitude = 1.49646,
                    altitude = 364, error = 10)

##(2a) calculate cosmic dose rate (two absorber)
calc_CosmicDoseRate(depth = c(5.0, 2.78), density = c(2.65, 1.7),
                    latitude = 38.06451, longitude = 1.49646,
                    altitude = 364, error = 10)

##(2b) calculate cosmic dose rate (two absorber) and
##correct for geomagnetic field changes
calc_CosmicDoseRate(depth = c(5.0, 2.78), density = c(2.65, 1.7),
                    latitude = 12.04332, longitude = 4.43243,
                    altitude = 364, corr.fieldChanges = TRUE,
```

```
                    est.age = 67, error = 15)


##(3) calculate cosmic dose rate and export results to .csv file
#calculate cosmic dose rate and save to variable
results<- calc_CosmicDoseRate(depth = 2.78, density = 1.7,
                              latitude = 38.06451, longitude = 1.49646,
                              altitude = 364, error = 10)

# the results can be accessed by
get_RLum.Results(results, "results")

#export results to .csv file - uncomment for usage
#write.csv(results, file = "c:/users/public/results.csv")

##(4) calculate cosmic dose rate for 6 samples from the same profile
##    and save to .csv file
#calculate cosmic dose rate and save to variable
results<- calc_CosmicDoseRate(depth = c(0.1, 0.5 , 2.1, 2.7, 4.2, 6.3),
                              density = 1.7, latitude = 38.06451,
                              longitude = 1.49646, altitude = 364,
                              error = 10)

#export results to .csv file - uncomment for usage
#write.csv(results, file = "c:/users/public/results_profile.csv")
```

`calc_FadingCorr` *Apply a fading correction according to Huntley & Lamothe (2001) for a given g-value.*

Description

This function runs the iterations that are needed to calculate the corrected age including the error for a given g-value according to Huntley & Lamothe (2001).

Usage

```
calc_FadingCorr(g_value, tc, age.faded, n.MCruns = 500)
```

Arguments

`g_value`	vector (**required**): g-value and error obtained from separate fading measurements (see example)
`tc`	numeric (**required**): time in seconds (time between irradiation and the prompt measurement, cf. Huntely & Lamothe 2001)
`age.faded`	numeric vector (**required**): uncorrected age with error in ka (see example)
`n.MCruns`	integer (with default): number of Monte Carlo simulation runs for error estimation

Details

The error of the fading-corrected age is determined using a Monte Carlo simulation approach. Large values for `n.MCruns` will significantly increase the computation time.

Value

A data.frame containing the fading-corrected age is returned.

Function version

0.1.2 (2014-04-13 14:27:43)

Note

The upper age limit is set to 500 ka!

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany) R Luminescence Package Team

References

Huntley, D.J., Lamothe, M., 2001. Ubiquity of anomalous fading in K-feldspars and the measurement and correction for it in optical dating. Canadian Journal of Earth Sciences, 38, 1093-1106.

See Also

#

Examples

```
calc_FadingCorr(g_value = c(3.3,0.03), tc = 752,
                age.faded = c(100,10),
                n.MCruns=50)
```

---

calc_FiniteMixture *Apply the finite mixture model (FMM) after Galbraith (2005) to a given De distribution*

Description

This function fits a k-component mixture to a De distribution with differing known standard errors. Parameters (doses and mixing proportions) are estimated by maximum likelihood assuming that the log dose estimates are from a mixture of normal distributions.

Usage

```
calc_FiniteMixture(input.data, sigmab, n.components, sample.id = "unknown sample",
   n.iterations = 200, grain.probability = FALSE, main = "Finite Mixture Model",
    dose.scale, pdf.weight = TRUE, pdf.sigma = "sigmab", pdf.colors = "gray",
    pdf.scale, plot.proportions = TRUE)
```

Arguments

input.data	RLum.Results or data.frame (**required**): for data.frame: two columns with De (input.data[,1]) and De error (values[,2])
sigmab	numeric (**required**): spread in De values given as a fraction (e.g. 0.2). This value represents the expected overdispersion in the data should the sample be well-bleached (Cunningham & Wallinga 2012, p. 100).
n.components	numeric (**required**): number of components to be fitted. If a vector is provided (e.g. c(2:8)) the finite mixtures for 2, 3 ... 8 components are calculated and a plot and a statistical evaluation of the model performance (BIC score and maximum log-likelihood) is provided.
sample.id	character (with default): sample id
n.iterations	numeric (with default): number of iterations for maximum log likelihood estimates
grain.probability	logical (with default): prints the estimated probabilities of which component each grain is in
main	character (with default): plot main title
dose.scale	numeric: manually set the scaling of the y-axis of the first plot with a vector in the form of c(min,max)
pdf.weight	logical (with default): weight the probability density functions by the components proportion (applies only when a vector is provided for n.components)
pdf.sigma	character (with default): if "sigmab" the components normal distributions are plotted with a common standard deviation (i.e. sigmab) as assumed by the FFM. Alternatively, "se" takes the standard error of each component for the sigma parameter of the normal distribution
pdf.colors	character (with default): color coding of the components in the the plot. Possible options are "gray", "colors" and "none"
pdf.scale	numeric: manually set the max density value for proper scaling of the x-axis of the first plot
plot.proportions	logical (with default): plot barplot showing the proportions of components

Details

This model uses the maximum likelihood and Bayesian Information Criterion (BIC) approaches.

Indications of overfitting are:

- increasing BIC
- repeated dose estimates
- covariance matrix not positive definite
- covariance matrix produces NaNs
- convergence problems

Plot

If a vector (`c(k.min:k.max)`) is provided for `n.components` a plot is generated showing the the k components equivalent doses as normal distributions. By default `pdf.weight` is set to `FALSE`, so that the area under each normal distribution is always 1. If `TRUE`, the probability density functions are weighted by the components proportion for each iteration of k components, so the sum of areas of each component equals 1. While the density values are on the same scale when no weights are used, the y-axis are individually scaled if the probability density are weighted by the components proportion.
The standard deviation (sigma) of the normal distributions is by default determined by a common `sigmab` (see `pdf.sigma`). For `pdf.sigma = "se"` the standard error of each component is taken instead.
The stacked barplot shows the proportion of each component (in per cent) calculated by the FFM. The last plot shows the achieved BIC scores and maximum log-likelihood estimates for each iteration of k.

Value

Returns a terminal output. In addition a list is returned containing the following elements:

`mle.matrix`	matrix covariance matrix of maximum likelihood estimates.
`grain.probability`	matrix with estimated probabilities of which component each grain is in.
`meta`	data.frame containing model parameters (sample.id, sigmab, n.components, llik, bic).
`components`	data.frame containing fitted components.
`single.comp`	data.frame containing log likelihood and BIC for a single component.

If a vector for `n.components` is provided (e.g. `c(2:8)`), `mle.matrix, grain.probability` and `meta` are lists containing matrices of the results for each iteration of the model.

The output should be accessed using the function `get_RLum.Results`

Function version

0.32 (2014-05-09 17:14:59)

Author(s)

Christoph Burow, University of Cologne (Germany)
Based on a rewritten S script of Rex Galbraith, 2006.

R Luminescence Package Team

References

Galbraith, R.F. & Green, P.F., 1990. Estimating the component ages in a finite mixture. Nuclear Tracks and Radiation Measurements, 17, pp. 197-206.

Galbraith, R.F. & Laslett, G.M., 1993. Statistical models for mixed fission track ages. Nuclear Tracks Radiation Measurements, 4, pp. 459-470.

Galbraith, R.F. & Roberts, R.G., 2012. Statistical aspects of equivalent dose and error calculation and display in OSL dating: An overview and some recommendations. Quaternary Geochronology, 11, pp. 1-27.

Roberts, R.G., Galbraith, R.F., Yoshida, H., Laslett, G.M. & Olley, J.M., 2000. Distinguishing dose populations in sediment mixtures: a test of single-grain optical dating procedures using mixtures of laboratory-dosed quartz. Radiation Measurements, 32, pp. 459-465.

Galbraith, R.F., 2005. Statistics for Fission Track Analysis, Chapman & Hall/CRC, Boca Raton.

Further reading

Arnold, L.J. & Roberts, R.G., 2009. Stochastic modelling of multi-grain equivalent dose (De) distributions: Implications for OSL dating of sediment mixtures. Quaternary Geochronology, 4, pp. 204-230.

Cunningham, A.C. & Wallinga, J., 2012. Realizing the potential of fluvial archives using robust OSL chronologies. Quaternary Geochronology, 12, pp. 98-106.

Rodnight, H., Duller, G.A.T., Wintle, A.G. & Tooth, S., 2006. Assessing the reproducibility and accuracy of optical dating of fluvial deposits. Quaternary Geochronology, 1, pp. 109-120.

Rodnight, H. 2008. How many equivalent dose values are needed to obtain a reproducible distribution?. Ancient TL, 26, pp. 3-10.

See Also

calc_CentralDose, calc_CommonDose, calc_FuchsLang2001, calc_MinDose3, calc_MinDose4

Examples

```
## load example data
data(ExampleData.DeValues, envir = environment())

## (1) apply the finite mixture model
## NOTE: the data set is not suitable for the finite mixture model,
## which is why a very small sigmab is necessary
calc_FiniteMixture(ExampleData.DeValues,
                   sigmab = 0.08, n.components = 2,
                   grain.probability = TRUE)

## (2) repeat the finite mixture model for 2, 3 and 4 maximum number of fitted
```

```
## components and save results
## NOTE: The following example is computationally intensive. Please un-comment
## the following lines to make the example work.
#res<- calc_FiniteMixture(ExampleData.DeValues,
#                         sigmab = 0.01, n.components = c(2:4),
#                         pdf.weight = TRUE, dose.scale = c(2200,4500))

## show structure of the results
#res

## show the results on equivalent dose, standard error and proportion of
## fitted components
#get_RLum.Results(object=res, data.object="components")
```

`calc_FuchsLang2001` *Apply the model after Fuchs & Lang (2001) to a given De distribution.*

Description

This function applies the method according to Fuchs & Lang (2001) for heterogeneously bleached samples with a given coefficient of variation threshold.

Usage

```
calc_FuchsLang2001(sample, sample.mtext = "unknown sample", sample.id = sample.mtext,
   cvThreshold = 5, startDeValue = 1, output.plot = TRUE, output.terminal = TRUE,
    ...)
```

Arguments

`sample`	RLum.Results or data.frame (**required**): for `data.frame`: two columns with De (`input.data[,1]`) and De error (`values[,2]`)
`sample.mtext`	character (optional): mtext for optional plot (top)
`sample.id`	character (with default): sample id, with default the sample.mtext is used.
`cvThreshold`	numeric (with default): coefficient of variation in percent, as threshold for the method, e.g. `cvThreshold = 3`. See details.
`startDeValue`	numeric (with default): number of the first aliquot that is used for the calculations
`output.plot`	logical (with default): plot output `TRUE/FALSE`
`output.terminal`	logical (with default): terminal output `TRUE/FALSE`
`...`	further arguments and graphical parameters passed to `plot`

Details

Used values

If the coefficient of variation (c[v]) of the first two values is larger than the threshold c[v_threshold], the first value is skipped. Use the `startDeValue` argument to define a start value for calculation (e.g. 2nd or 3rd value).

Basic steps of the approach

(1) Estimate natural relative variation of the sample using a dose recovery test
(2) Sort the input values ascendingly
(3) Calculate a running mean, starting with the lowermost two values and add values iteratively.
(4) Stop if the calculated c[v] exceeds the specified `cvThreshold`

Value

A plot and terminal output is provided if desired. In addition, a list is returned containing two elements:

`results`	data.frame with stastical parameters, e.g. mean, sd, ...
`usedDeValues`	data.frame containing the used values for the calculation

Function version

0.4.0 (2014-04-13 14:27:55)

Note

Please consider the requirements and the constraints of this method (see Fuchs & Lang, 2001)

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany) R Luminescence Package Team

References

Fuchs, M. & Lang, A., 2001. OSL dating of coarse-grain fluvial quartz using single-aliqout protocols on sediments from NE Peloponnese, Greece. In: Quaternary Science Reviews, 20, 783-787.

Fuchs, M. & Wagner, G.A., 2003. Recognition of insufficient bleaching by small aliquots of quartz for reconstructing soil erosion in Greece. Quaternary Science Reviews, 22, 1161-1167.

See Also

`plot`, `calc_MinDose3`, `calc_MinDose4` `calc_FiniteMixture`, `calc_CentralDose`, `calc_CommonDose`, `RLum.Results`

Examples

```
##load example data
data(ExampleData.DeValues, envir = environment())

##calculate De according to Fuchs & Lang (2001)
temp <- calc_FuchsLang2001(ExampleData.DeValues, cvThreshold = 5)

##show values
get_RLum.Results(temp)
```

---

calc_HomogeneityTest *Apply a simple homogeneity test after Galbraith (2003)*

Description

A simple homogeneity test for De estimates

Usage

```
calc_HomogeneityTest(input.data, log = TRUE, sample.id = "unknown sample",
    ...)
```

Arguments

input.data	RLum.Results or data.frame (**required**): for data.frame: two columns with De (input.data[,1]) and De error (values[,2])
log	logical (with default): peform the homogeniety test with (un-)logged data
sample.id	character (with default): sample id
...	further arguments (for internal compatibility only).

Details

For details see Galbraith (2003).

Value

Returns a terminal output. In addition an RLum.Results object is returned containing the following element:

results	data.frame with statistical parameters.

The output should be accessed using the function get_RLum.Results

Function version

0.2 (2014-04-13 14:28:06)

Author(s)

Christoph Burow, University of Cologne (Germany),
R Luminescence Package Team

References

Galbraith, R.F., 2003. A simple homogeneity test for estimates of dose obtained using OSL. Ancient TL, 21, pp. 75-77.

See Also

pchisq

Examples

```
## load example data
data(ExampleData.DeValues, envir = environment())

## apply the homogeneity test
calc_HomogeneityTest(ExampleData.DeValues)
```

---

calc_MaxDose3 *Apply the maximum age model to a given De distribution*

Description

Function to fit the maximum age model to De data. This function is a modified version of the three parameter minimum age model after Galbraith et al. (1999) using a similiar approach described in Olley et al. (2006).

Usage

```
calc_MaxDose3(input.data, sigmab, log = TRUE, sample.id = "unknown sample",
    gamma.xlb = 0.1, gamma.xub = 100, sigma.xlb = 0.001, sigma.xub = 5,
    init.gamma = 10, init.sigma = 1.2, init.p0 = 0.01, ignore.NA = FALSE,
    calc.ProfileLikelihoods = TRUE, console.ProfileLikelihoods = FALSE,
    console.extendedOutput = FALSE, output.plot = FALSE, output.indices = 3)
```

Arguments

input.data	RLum.Results or data.frame (**required**): for data.frame: two columns with De (input.data[,1]) and De error (values[,2])
sigmab	numeric (**required**): spread in De values given as a fraction (e.g. 0.2). This value represents the expected overdispersion in the data should the sample be well-bleached (Cunningham & Walling 2012, p. 100).
log	logical (with default): fit the (un-)logged three parameter maximum dose model to De data. An un-logged version is currently not supported

`sample.id` `character` (with default): sample id

`gamma.xlb` `numeric` (with default): lower boundary of gamma

`gamma.xub` `numeric` (with default): upper boundary of gamma

`sigma.xlb` `numeric` (with default): lower boundary of sigma

`sigma.xub` `numeric` (with default): upper boundary of sigma

`init.gamma` `numeric` (with default): starting value of gamma

`init.sigma` `numeric` (with default): starting value of sigma

`init.p0` `numeric` (with default): starting value of p0

`ignore.NA` `logical` (with default): ignore NA values during log likelihood calculations. See details.

`calc.ProfileLikelihoods`
`logical` (with default): calculate profile log likelihood functions for gamma, sigma, p0. See `output.indices`.

`console.ProfileLikelihoods`
`logical` (with default): print profile log likelihood functions for gamma, sigma, p0 to console.

`console.extendedOutput`
`logical` (with default): extended terminal output

`output.plot` `logical` (with default): plot output (`TRUE`/`FALSE`)

`output.indices` `numeric` (with default): requires `calc.ProfileLikelihoods = TRUE`. Indices: 1 = gamma, 2 = gamma/sigma, 3 = gamma/sigma/p0.

Details

Parameters

This model has three parameters:

`gamma`:	maximum dose on the log scale
`sigma`:	spread in ages above the maximum
`p0`:	proportion of grains at gamma

Data transformation

To estimate the maximum dose population and its standard error, the three parameter minimum age model of Galbraith et al. (1999) is adapted. The measured De values are transformed as follows:

1. convert De values to natural logs
2. multiply the logged data to creat a mirror image of the De distribution
3. shift De values along x-axis by the smallest x-value found to obtain only positive values
4. combine in quadrature the measurement error associated with each De value with a relative error specified by sigmab 5. apply the MAM to these data

When all calculations are done the results are then converted as follows

1. subtract the x-offset
2. multiply the natural logs by -1
3. take the exponent to obtain the maximum dose estimate in Gy

(Un-)logged model

In the original version of the three-parameter maximum dose model, the basic data are the natural logarithms of the De estimates and relative standard errors of the De estimates. This model will be applied if `log = TRUE`.

If `log = FALSE`, the modified un-logged model will be applied instead. This has essentially the same form as the original version. `gamma` and `sigma` are in Gy and `gamma` becomes the maximum true dose in the population. NOTE: This option is disabled for the maximum dose model, as the data transformation requires logged De values!

While the original (logged) version of the mimimum dose model may be appropriate for most samples (i.e. De distributions), the modified (un-logged) version is specially designed for modern-age and young samples containing negative, zero or near-zero De estimates (Arnold et al. 2009, p. 323).

Boundaries

Depending on the data, the upper and lower bounds for gamma (`gamma.xlb` and `gamma.xub`) need to be specified. If the final estimate of gamma is on the boundary, `gamma.xlb` and `gamma.xub` need to be adjusted appropriately, so that gamma lies within the bounds. The same applies for sigma boundaries (`sigma.xlb` and `sigma.xub`).

Initial values

The log likelihood calculations use the nlminb function. Accordingly, initial values for the three parameters `init.gamma`, `init.sigma` and `init.p0` need to be specified.

Ignore NA values

In some cases during the calculation of the log likelihoods NA values are produced instantly terminating the minimum age model. It is advised to adjust some of the values provided for any argument. If the model still produces NA values it is possible to omit these values by setting `ignore.NA = TRUE`. While the model is then usually able to finish all calculations the integrity of the final estimates cannot be ensured. Use this argument at own risk.

Value

Returns a plot (optional) and terminal output. In addition an `RLum.Results` object is returned containing the following element:

`results` data.frame with statistical parameters.

The output should be accessed using the function get_RLum.Results

Function version

0.22 (2014-04-13 14:28:21)

Note

The default boundary and starting values for *gamma*, *sigma* and *p0* may only be appropriate for some De data sets and may need to be changed for other data. An un-logged version is currently not supported.

Author(s)

Christoph Burow, University of Cologne (Germany)
Based on a rewritten S script of Rex Galbraith, 2010

R Luminescence Package Team

References

Arnold, L.J., Roberts, R.G., Galbraith, R.F. & DeLong, S.B., 2009. A revised burial dose estimation procedure for optical dating of young and modern-age sediments. Quaternary Geochronology, 4, pp. 306-325.

Galbraith, R.F. & Laslett, G.M., 1993. Statistical models for mixed fission track ages. Nuclear Tracks Radiation Measurements, 4, pp. 459-470.

Galbraith, R.F., Roberts, R.G., Laslett, G.M., Yoshida, H. & Olley, J.M., 1999. Optical dating of single grains of quartz from Jinmium rock shelter, northern Australia. Part I: experimental design and statistical models. Archaeometry, 41, pp. 339-364.

Galbraith, R.F., 2005. Statistics for Fission Track Analysis, Chapman & Hall/CRC, Boca Raton.

Galbraith, R.F. & Roberts, R.G., 2012. Statistical aspects of equivalent dose and error calculation and display in OSL dating: An overview and some recommendations. Quaternary Geochronology, 11, pp. 1-27.

Olley, J.M., Roberts, R.G., Yoshida, H., Bowler, J.M., 2006. Single-grain optical dating of grave-infill associated with human burials at Lake Mungo, Australia. Quaternary Science Reviews, 25, pp. 2469-2474.

Further reading

Arnold, L.J. & Roberts, R.G., 2009. Stochastic modelling of multi-grain equivalent dose (De) distributions: Implications for OSL dating of sediment mixtures. Quaternary Geochronology, 4, pp. 204-230.

Bailey, R.M. & Arnold, L.J., 2006. Statistical modelling of single grain quartz De distributions

and an assessment of procedures for estimating burial dose. Quaternary Science Reviews, 25, pp. 2475-2502.

Cunningham, A.C. & Wallinga, J., 2012. Realizing the potential of fluvial archives using robust OSL chronologies. Quaternary Geochronology, 12, pp. 98-106.

Rodnight, H., Duller, G.A.T., Wintle, A.G. & Tooth, S., 2006. Assessing the reproducibility and accuracy of optical dating of fluvial deposits. Quaternary Geochronology, 1, pp. 109-120.

Rodnight, H., 2008. How many equivalent dose values are needed to obtain a reproducible distribution?. Ancient TL, 26, pp. 3-10.

See Also

nlminb, calc_CentralDose, calc_CommonDose, calc_FiniteMixture, calc_FuchsLang2001, calc_MinDose4

Examples

```
## load example data
data(ExampleData.DeValues, envir = environment())

## apply the logged maximum dose model
## NOTE THAT THE EXAMPLE DATA SET IS NOT SUITABLE FOR THE
## MAXIMUM DOSE MODEL.
calc_MaxDose3(ExampleData.DeValues,
              sigmab = 0.3, gamma.xub = 4000)
```

---

calc_MinDose3 *Apply the (un-)logged three parameter minimum age model (MAM 3) after Galbraith et al. (1999) to a given De distribution*

Description

Function to fit the (un-)logged three parameter minimum dose model (MAM 3) to De data.

Usage

```
calc_MinDose3(input.data, sigmab, log = TRUE, sample.id = "unknown sample",
    gamma.xlb = 0.1, gamma.xub = 100, sigma.xlb = 0.001, sigma.xub = 5,
    init.gamma = 10, init.sigma = 1.2, init.p0 = 0.01, ignore.NA = FALSE,
    calc.ProfileLikelihoods = TRUE, console.ProfileLikelihoods = FALSE,
    console.extendedOutput = FALSE, output.plot = TRUE, output.indices = 3)
```

Arguments

input.data	RLum.Results or data.frame (**required**): for data.frame: two columns with De (input.data[,1]) and De error (values[,2])
sigmab	numeric (**required**): spread in De values given as a fraction (e.g. 0.2). This value represents the expected overdispersion in the data should the sample be well-bleached (Cunningham & Walling 2012, p. 100).
log	logical (with default): fit the (un-)logged three parameter minimum dose model to De data
sample.id	character (with default): sample id
gamma.xlb	numeric (with default): lower boundary of gamma
gamma.xub	numeric (with default): upper boundary of gamma
sigma.xlb	numeric (with default): lower boundary of sigma
sigma.xub	numeric (with default): upper boundary of sigma
init.gamma	numeric (with default): starting value of gamma
init.sigma	numeric (with default): starting value of sigma
init.p0	numeric (with default): starting value of p0
ignore.NA	logical (with default): ignore NA values during log likelihood calculations. See details.
calc.ProfileLikelihoods	logical (with default): calculate profile log likelihood functions for gamma, sigma, p0. See output.indices.
console.ProfileLikelihoods	logical (with default): print profile log likelihood functions for gamma, sigma, p0 to console.
console.extendedOutput	logical (with default): extended terminal output
output.plot	logical (with default): plot output (TRUE/FALSE)
output.indices	numeric (with default): requires calc.ProfileLikelihoods = TRUE. Indices: 1 = gamma, 2 = gamma/sigma, 3 = gamma/sigma/p0.

Details

Parameters

This model has three parameters:

gamma:	minimum dose on the log scale
sigma:	spread in ages above the minimum
p0:	proportion of grains at gamma

(Un-)logged model

In the original version of the three-parameter minimum dose model, the basic data are the natural logarithms of the De estimates and relative standard errors of the De estimates. This model will be applied if `log = TRUE`.

If `log = FALSE`, the modified un-logged model will be applied instead. This has essentially the same form as the original version. `gamma` and `sigma` are in Gy and `gamma` becomes the minimum true dose in the population.

While the original (logged) version of the mimimum dose model may be appropriate for most samples (i.e. De distributions), the modified (un-logged) version is specially designed for modern-age and young samples containing negative, zero or near-zero De estimates (Arnold et al. 2009, p. 323).

Boundaries

Depending on the data, the upper and lower bounds for gamma (`gamma.xlb` and `gamma.xub`) need to be specified. If the final estimate of gamma is on the boundary, `gamma.xlb` and `gamma.xub` need to be adjusted appropriately, so that gamma lies within the bounds. The same applies for sigma boundaries (`sigma.xlb` and `sigma.xub`).

Initial values

The log likelihood calculations use the nlminb function. Accordingly, initial values for the three parameters `init.gamma`, `init.sigma` and `init.p0` need to be specified.

Ignore NA values

In some cases during the calculation of the log likelihoods NA values are produced instantly terminating the minimum age model. It is advised to adjust some of the values provided for any argument. If the model still produces NA values it is possible to omit these values by setting `ignore.NA = TRUE`. While the model is then usually able to finish all calculations the integrity of the final estimates cannot be ensured. Use this argument at own risk.

Value

Returns a plot (optional) and terminal output. In addition an `RLum.Results` object is returned containing the following element:

`results` data.frame with statistical parameters.

The output should be accessed using the function `get_RLum.Results`

Function version

0.22 (2014-04-13 14:28:27)

Note

The default boundary and starting values for *gamma*, *sigma* and *p0* may only be appropriate for some De data sets and may need to be changed for other data. This is especially true when the

un-logged version is applied.

Author(s)

Christoph Burow, University of Cologne (Germany)
Based on a rewritten S script of Rex Galbraith, 2010

R Luminescence Package Team

References

Arnold, L.J., Roberts, R.G., Galbraith, R.F. & DeLong, S.B., 2009. A revised burial dose estimation procedure for optical dating of young and modern-age sediments. Quaternary Geochronology, 4, pp. 306-325.

Galbraith, R.F. & Laslett, G.M., 1993. Statistical models for mixed fission track ages. Nuclear Tracks Radiation Measurements, 4, pp. 459-470.

Galbraith, R.F., Roberts, R.G., Laslett, G.M., Yoshida, H. & Olley, J.M., 1999. Optical dating of single grains of quartz from Jinmium rock shelter, northern Australia. Part I: experimental design and statistical models. Archaeometry, 41, pp. 339-364.

Galbraith, R.F., 2005. Statistics for Fission Track Analysis, Chapman & Hall/CRC, Boca Raton.

Galbraith, R.F. & Roberts, R.G., 2012. Statistical aspects of equivalent dose and error calculation and display in OSL dating: An overview and some recommendations. Quaternary Geochronology, 11, pp. 1-27.

Further reading

Arnold, L.J. & Roberts, R.G., 2009. Stochastic modelling of multi-grain equivalent dose (De) distributions: Implications for OSL dating of sediment mixtures. Quaternary Geochronology, 4, pp. 204-230.

Bailey, R.M. & Arnold, L.J., 2006. Statistical modelling of single grain quartz De distributions and an assessment of procedures for estimating burial dose. Quaternary Science Reviews, 25, pp. 2475-2502.

Cunningham, A.C. & Wallinga, J., 2012. Realizing the potential of fluvial archives using robust OSL chronologies. Quaternary Geochronology, 12, pp. 98-106.

Rodnight, H., Duller, G.A.T., Wintle, A.G. & Tooth, S., 2006. Assessing the reproducibility and accuracy of optical dating of fluvial deposits. Quaternary Geochronology, 1, pp. 109-120.

Rodnight, H., 2008. How many equivalent dose values are needed to obtain a reproducible distribution?. Ancient TL, 26, pp. 3-10.

See Also

nlminb, calc_CentralDose, calc_CommonDose, calc_FiniteMixture, calc_FuchsLang2001, calc_MinDose4

Examples

```
## load example data
data(ExampleData.DeValues, envir = environment())

## apply the logged minimum dose model
calc_MinDose3(ExampleData.DeValues,
              sigmab = 0.3,gamma.xub = 7000,
              output.plot = FALSE)

## apply the un-logged minimum dose model
## note that the example data set does not meet the un-logged model
## requirements
calc_MinDose3(ExampleData.DeValues, log = FALSE,
              sigmab = 0.3,gamma.xub = 5000,
              output.plot = FALSE)
```

---

calc_MinDose4 *Apply the (un-)logged four parameter minimum age model (MAM 4) after Galbraith et al. (1999) to a given De distribution*

Description

Function to fit the (un-)logged four parameter minimum dose model (MAM 4) to De data.

Usage

```
calc_MinDose4(input.data, sigmab, log = TRUE, sample.id = "unknown sample",
    gamma.xlb = 0.1, gamma.xub = 100, mu.xlb = 1, mu.xub = 100,
    sigma.xlb = 0.001, sigma.xub = 5, init.gamma = 10, init.mu = 10,
   init.sigma = 0.6, init.p0 = 0.01, ignore.NA = FALSE, calc.ProfileLikelihoods = TRUE,
    console.ProfileLikelihoods = FALSE, console.extendedOutput = FALSE,
    output.plot = TRUE, output.indices = 4)
```

Arguments

input.data	RLum.Results or data.frame (**required**): for data.frame: two columns with De (input.data[,1]) and De error (values[,2])
sigmab	numeric (**required**): spread in De values given as a fraction (e.g. 0.2). This value represents the expected overdispersion in the data should the sample be well-bleached (Cunningham & Walling 2012, p. 100).
log	logical (with default): fit the (un-)logged three parameter minimum dose model to De data

sample.id	character (with default): sample id
gamma.xlb	numeric (with default): lower boundary of gamma
gamma.xub	numeric (with default): upper boundary of gamma
mu.xlb	numeric (with default): lower boundary of mu
mu.xub	numeric (with default): upper boundary of mu
sigma.xlb	numeric (with default): lower boundary of sigma
sigma.xub	numeric (with default): upper boundary of sigma
init.gamma	numeric (with default): starting value of gamma
init.mu	numeric (with default): starting value of mu
init.sigma	numeric (with default): starting value of sigma
init.p0	numeric (with default): starting value of p0
ignore.NA	logical (with default): ignore NA values during log likelihood calculations. See details.
calc.ProfileLikelihoods	logical (with default): calculate profile log likelihood functions for gamma, mu, sigma, p0. See output.indices.
console.ProfileLikelihoods	logical (with default): print profile log likelihood functions for gamma, mu, sigma, p0 to console.
console.extendedOutput	logical (with default): extended terminal output
output.plot	logical (with default): plot output (TRUE/FALSE)
output.indices	numeric (with default): requires calc.ProfileLikelihoods = TRUE. Indices: 1 = gamma, 2 = gamma/mu, 3 = gamma/mu/sigma, 4 = gamma/mu/sigma/p0

Details

Parameters

This model has four parameters:

gamma:	minimum dose on the log scale
mu:	mean of the non-truncated normal distribution
sigma:	spread in ages above the minimum
p0:	proportion of grains at gamma

(Un-)logged model

In the original version of the three-parameter minimum dose model, the basic data are the natural logarithms of the De estimates and relative standard errors of the De estimates. This model will be applied if log = TRUE.

If `log = FALSE`, the modified un-logged model will be applied instead. This has essentially the same form as the original version. `gamma` and `sigma` are in Gy and `gamma` becomes the minimum true dose in the population.

While the original (logged) version of the mimimum dose model may be appropriate for most samples (i.e. De distributions), the modified (un-logged) version is specially designed for modern-age and young samples containing negative, zero or near-zero De estimates (Arnold et al. 2009, p. 323).

Boundaries

Depending on the data, the upper and lower bounds for gamma (`gamma.xlb` and `gamma.xub`) and mu (`mu.xlb` and `mu.xub`) need to be specified. If the final estimate of gamma or mu is on the boundary, `gamma.xlb` and `gamma.xub` (`mu.xlb` and `mu.xub` respectively) need to be adjusted appropriately, so that gamma and mu lie within the bounds. The same applies for sigma boundaries (`sigma.xlb` and `sigma.xub`)

Initial values

The log likelihood calculations use the nlminb function. Accordingly, initial values for the four parameters `init.gamma`, `init.sigma`, `init.mu` and `init.p0` need to be specified.

Ignore NA values

In some cases during the calculation of the log likelihoods NA values are produced instantly terminating the minimum age model. It is advised to adjust some of the values provided for any argument. If the model still produces NA values it is possible to omit these values by setting `ignore.NA = TRUE`. While the model is then usually able to finish all calculations the integrity of the final estimates cannot be ensured. Use this argument at own risk.

Value

Returns a plot (optional) and terminal output. A file containing statistical results is provided if desired. In addition an `RLum.Results` object is returned containing the following element:

`results` data.frame with statistical parameters.

The output should be accessed using the function `get_RLum.Results`

Function version

0.22 (2014-04-13 14:28:35)

Note

The default boundary and starting values for *gamma*, *mu*, *sigma* and *p0* may only be appropriate for some De data sets and may need to be changed for other data. This is especially true when the un-logged version is applied.

Author(s)

Christoph Burow, University of Cologne (Germany)
Based on a rewritten S script of Rex Galbraith, 2010

R Luminescence Package Team

References

Arnold, L.J., Roberts, R.G., Galbraith, R.F. & DeLong, S.B., 2009. A revised burial dose estimation procedure for optical dating of young and modern-age sediments. Quaternary Geochronology, 4, pp. 306-325.

Galbraith, R.F. & Laslett, G.M., 1993. Statistical models for mixed fission track ages. Nuclear Tracks Radiation Measurements, 4, pp. 459-470.

Galbraith, R.F., Roberts, R.G., Laslett, G.M., Yoshida, H. & Olley, J.M., 1999. Optical dating of single grains of quartz from Jinmium rock shelter, northern Australia. Part I: experimental design and statistical models. Archaeometry, 41, pp. 339-364.

Galbraith, R.F., 2005. Statistics for Fission Track Analysis, Chapman & Hall/CRC, Boca Raton.

Galbraith, R.F. & Roberts, R.G., 2012. Statistical aspects of equivalent dose and error calculation and display in OSL dating: An overview and some recommendations. Quaternary Geochronology, 11, pp. 1-27.

Further reading

Arnold, L.J. & Roberts, R.G., 2009. Stochastic modelling of multi-grain equivalent dose (De) distributions: Implications for OSL dating of sediment mixtures. Quaternary Geochronology, 4, pp. 204-230.

Bailey, R.M. & Arnold, L.J., 2006. Statistical modelling of single grain quartz De distributions and an assessment of procedures for estimating burial dose. Quaternary Science Reviews, 25, pp. 2475-2502.

Cunningham, A.C. & Wallinga, J., 2012. Realizing the potential of fluvial archives using robust OSL chronologies. Quaternary Geochronology, 12, pp. 98-106.

Rodnight, H., Duller, G.A.T., Wintle, A.G. & Tooth, S., 2006. Assessing the reproducibility and accuracy of optical dating of fluvial deposits. Quaternary Geochronology, 1, pp. 109-120.

Rodnight, H., 2008. How many equivalent dose values are needed to obtain a reproducible distribution?. Ancient TL, 26, pp. 3-10.

See Also

nlminb, calc_CentralDose, calc_CommonDose, calc_FiniteMixture, calc_FuchsLang2001, calc_MinDose3

Examples

```
## load example data
data(ExampleData.DeValues, envir = environment())

## apply the logged minimum dose model
calc_MinDose4(ExampleData.DeValues,
              sigmab = 0.05, gamma.xub = 10000, mu.xub = 10000, init.p0 = 0.4,
              output.plot = FALSE)
```

---

calc_OSLLxTxRatio *Calculate Lx/Tx ratio for CW-OSL curves.*

Description

Calculate Lx/Tx ratios from a given set of CW-OSL curves.

Usage

```
calc_OSLLxTxRatio(Lx.data, Tx.data, signal.integral, background.integral,
    background.count.distribution = "non-poisson", sigmab)
```

Arguments

Lx.data	data.frame (**required**): requires a CW-OSL shine down curve (x = time, y = counts)
Tx.data	data.frame (optional): requires a CW-OSL shine down curve (x = time, y = counts). If no input is given the Tx.data will be treated as NA and no Lx/Tx ratio is calculated.
signal.integral	vector (**required**): vector with the limits for the signal integral.
background.integral	vector (**required**): vector with the bounds for the background integral.
background.count.distribution	character (with default): Sets the count distribution assumed for the error calculation. Possible arguments poisson or non-poisson. See details for further information
sigmab	numeric (optional): Option to set a manual value for the overdispersion, used for the Lx/Tx error calculation. The value should be provided as absolute squared count values.

Details

The integrity of the chosen values for the signal and background integral is checked by the function; the signal integral limits have to be lower than the background integral limits. If a vector is given as input instead of a data.frame, an artificial `data.frame` is produced. The error calculation is done according to Galbraith (2002).

background.count.distribution

This argument allows selecting the distribution assumption that is used for the error calculation. According to Galbraith (2002, 2014) the background counts may be overdispersed (i.e. do not follow a poisson distribution, which is assumed for the photomultiplier counts). In that case (might be the normal case) it has to be accounted for the overdispersion by estimating σ^2 (i.e. the overdispersion value). Therefore the relative standard error is calculated as:

(a) `poisson`

$$rse(\mu_S) \approx \sqrt{(Y_0 + Y_1/k^2)/Y_0 - Y_1/k}$$

(b) `non-poisson`

$$rse(\mu_S) \approx \sqrt{(Y_0 + Y_1/k^2 + \sigma^2(1 + 1/k))/Y_0 - Y_1/k}$$

Value

Returns an S4 object of type `RLum.Results`. Slot `data` contains a list with the following structure:
$ LnLx
$ LnLx.BG
$ TnTx
$ TnTx.BG
$ Net_LnLx
$ Net_LnLx.Error
$ Net_TnTx.Error
$ LxTx
$ LxTx.Error

Function version

0.4.1 (2014-04-13 14:28:43)

Note

The results of this function have been cross-checked with the Analyst (vers. 3.24b). Access to the results object via `get_RLum.Results`.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

References

Duller, G., 2007. Analyst. http://www.nutech.dtu.dk/english/~/media/Andre_Universitetsenheder/Nutech/Produkter%20og%20services/Dosimetri/radiation_measurement_instruments/tl_osl_reader/Manuals/analyst_manual_v3_22b.ashx

Galbraith, R.F., 2002. A note on the variance of a background-corrected OSL count. Ancient TL, 20 (2), 49-51.

Galbraith, R.F., 2014. A further note on the variance of a background-corrected OSL count. Submitted to Ancient TL.

See Also

Analyse_SAR.OSLdata, plot_GrowthCurve, analyse_SAR.CWOSL

Examples

```
##load data
data(ExampleData.LxTxOSLData, envir = environment())

##calculate Lx/Tx ratio
results <- calc_OSLLxTxRatio(Lx.data, Tx.data, signal.integral = c(1:2),
                             background.integral = c(85:100))

##get results object
get_RLum.Results(results)
```

---

calc_Statistics *Function to calculate statistic measures*

Description

This function calculates a number of descriptive statistics for De-data, most fundamentally using error-weighted approaches.

Usage

```
calc_Statistics(data, weight.calc = "reciprocal")
```

Arguments

data	data.frame or RLum.Results object (required): for data.frame two columns: De (data[,1]) and De error (data[,2]). To plot several data sets in one plot the data sets must be provided as list, e.g. list(data.1, data.2).
weight.calc	character: type of weight calculation. One out of "reciprocal" (weight is 1/error), "square" (weight is 1/error^2).

Value

Returns a list with weighted and unweighted statistic measures.

Function version

0.1 (2014-04-13 14:28:51)

Author(s)

Michael Dietze, GFZ Potsdam (Germany),
R Luminescence Package Team

Examples

```
## load example data
data(ExampleData.DeValues, envir = environment())

## show a rough plot of the data to illustrate the non-normal distribution
plot_KDE(ExampleData.DeValues)

## calculate statistics and show output
str(calc_Statistics(ExampleData.DeValues))

## now the same for 10000 normal distributed random numbers with equal errors
x <- as.data.frame(cbind(rnorm(n = 10^5, mean = 0, sd = 1),
                         rep(0.001, 10^5)))

## note the congruent results for weighted and unweighted measures
str(calc_Statistics(x))
```

---

calc_TLLxTxRatio *Calculate the Lx/Tx ratio for a given set of TL curves [beta version]*

Description

Calculate Lx/Tx ratio for a given set of TL curves.

Usage

```
calc_TLLxTxRatio(Lx.data.signal, Lx.data.background, Tx.data.signal,
    Tx.data.background, signal.integral.min, signal.integral.max)
```

Arguments

`Lx.data.signal` data.frame (**required**): TL data (x = temperature, y = counts) (TL signal)

`Lx.data.background`
data.frame (optional): TL data (x = temperature, y = counts). If no data are provided no background subtraction is performed.

`Tx.data.signal` data.frame (**required**): TL data (x = temperature, y = counts) (TL test signal)

`Tx.data.background`
data.frame (optional): TL data (x = temperature, y = counts). If no data are provided no background subtraction is performed.

`signal.integral.min`
integer (**required**): channel number for the upper signal integral bound (e.g. `signal.integral.min = 100`)

`signal.integral.max`
integer (**required**): channel number for the upper signal integral bound (e.g. `signal.integral.max = 200`)

Details

-

Value

Returns an S4 object of type `RLum.Results`. Slot `data` contains a data.frame with the following structure:

$ LnLx
$ LnLx.BG
$ TnTx
$ TnTx.BG
$ Net_LnLx
$ Net_LnLx.Error

Function version

0.2.1 (2014-04-13 14:29:01)

Note

This function is a beta version!

Author(s)

Sebastian Kreutzer, JLU Giessen/Freiberg Instruments (Germany), Christoph Schmidt, University of Bayreuth (Germany),
R Luminescence Package Team

References

-

See Also

`RLum.Results`, `analyse_SAR.TL`

Examples

```
##load package example data
data(ExampleData.BINfileData, envir = environment())

##convert Risoe.BINfileData into a curve object
temp <- Risoe.BINfileData2RLum.Analysis(TL.SAR.Data, pos = 3)


Lx.data.signal <- get_RLum.Analysis(temp, record.id=1)
Lx.data.background <- get_RLum.Analysis(temp, record.id=2)
Tx.data.signal <- get_RLum.Analysis(temp, record.id=3)
Tx.data.background <- get_RLum.Analysis(temp, record.id=4)
signal.integral.min <- 210
signal.integral.max <- 230

output <- calc_TLLxTxRatio(Lx.data.signal,
                           Lx.data.background,
                           Tx.data.signal, Tx.data.background,
                           signal.integral.min, signal.integral.max)
get_RLum.Results(output)
```

CW2pHMi *Transform a CW-OSL curve into a pHM-OSL curve via interpolation under hyperbolic modulation conditions*

Description

This function transforms a conventionally measured continuous-wave (CW) OSL-curve to a pseudo hyperbolic modulated (pHM) curve under hyperbolic modulation conditions using the interpolation procedure described by Bos & Wallinga (2012).

Usage

```
CW2pHMi(values, delta)
```

Arguments

values	RLum.Data.Curve or data.frame (**required**): RLum.Data.Curve or data.frame with measured curve data of type stimulation time (t) (values[,1]) and measured counts (cts) (values[,2]).
delta	vector (optional): stimulation rate parameter, if no value is given, the optimal value is estimated automatically (see details). Smaller values of delta produce more points in the rising tail of the curve.

Details

The complete procedure of the transformation is described in Bos & Wallinga (2012). The input `data.frame` consists of two columns: time (t) and count values (CW(t))

Internal transformation steps

(1) log(CW-OSL) values

(2) Calculate t' which is the transformed time:

$$t' = t - (1/\delta) * log(1 + \delta * t)$$

(3) Interpolate CW(t'), i.e. use the log(CW(t)) to obtain the count values for the transformed time (t'). Values beyond `min(t)` and `max(t)` produce `NA` values.

(4) Select all values for t' < `min(t)`, i.e. values beyond the time resolution of t. Select the first two values of the transformed data set which contain no `NA` values and use these values for a linear fit using `lm`.

(5) Extrapolate values for t' < `min(t)` based on the previously obtained fit parameters.

(6) Transform values using

$$pHM(t) = (\delta * t/(1 + \delta * t)) * c * CW(t')$$

$$c = (1 + \delta * P)/\delta * P$$

$$P = length(stimulation \tilde{} period)$$

(7) Combine all values and truncate all values for t' > `max(t)`

The number of values for t' < `min(t)` *depends on the stimulation rate parameter* `delta`*. To avoid the production of too many artificial data at the raising tail of the determined pHM curve, it is recommended to use the automatic estimation routine for* `delta`*, i.e. provide no value for* `delta`*.*

Value

The function returns the same data type as the input data type with the transformed curve values.

`RLum.Data.Curve`
package `RLum` `object` with two additional info elements:

$CW2pHMi.x.t	: transformed time values
$CW2pHMi.method	: used method for the production of the new data points

`data.frame` with four columns:

$x	: time
$y.t	: transformed count values
$x.t	: transformed time values

$method : used method for the production of the new data points

Function version

0.2.1 (2014-04-13 14:29:09)

Note

According to Bos & Wallinga (2012), the number of extrapolated points should be limited to avoid artificial intensity data. If `delta` is provided manually and more than two points are extrapolated, a warning message is returned.

The function `approx` may produce some `Inf` and `NaN` data. The function tries to manually interpolate these values by calculating the `mean` using the adjacent channels. If two invalid values are succeeding, the values are removed and no further interpolation is attempted.
In every case a warning message is shown.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany)

Based on comments and suggestions from:
Adrie J.J. Bos, Delft University of Technology, The Netherlands

R Luminescence Package Team

References

Bos, A.J.J. & Wallinga, J., 2012. How to visualize quartz OSL signal components. Radiation Measurements, 47, 752-758.

Further Reading

Bulur, E., 1996. An Alternative Technique For Optically Stimulated Luminescence (OSL) Experiment. Radiation Measurements, 26, 701-709.

Bulur, E., 2000. A simple transformation for converting CW-OSL curves to LM-OSL curves. Radiation Measurements, 32, 141-145.

See Also

`CW2pLM`, `CW2pLMi`, `CW2pPMi`, `fit_LMCurve`, `lm`, `RLum.Data.Curve`

Examples

```
##(1) - simple transformation

##load CW-OSL curve data
data(ExampleData.CW_OSL_Curve, envir = environment())
```

```
##transform values
values.transformed<-CW2pHMi(ExampleData.CW_OSL_Curve)

##plot
plot(values.transformed$x, values.transformed$y.t, log = "x")

##(2) - load CW-OSL curve from BIN-file and plot transformed values

##load BINfile
#BINfileData<-readBIN2R("[path to BIN-file]")
data(ExampleData.BINfileData, envir = environment())

##grep first CW-OSL curve from ALQ 1

curve.ID<-CWOSL.SAR.Data@METADATA[CWOSL.SAR.Data@METADATA[,"LTYPE"]=="OSL" &
                                    CWOSL.SAR.Data@METADATA[,"POSITION"]==1
                                  ,"ID"]

curve.HIGH<-CWOSL.SAR.Data@METADATA[CWOSL.SAR.Data@METADATA[,"ID"]==curve.ID[1]
                                    ,"HIGH"]

curve.NPOINTS<-CWOSL.SAR.Data@METADATA[CWOSL.SAR.Data@METADATA[,"ID"]==curve.ID[1]
                                       ,"NPOINTS"]

##combine curve to data set

curve<-data.frame(x = seq(curve.HIGH/curve.NPOINTS,curve.HIGH,
                          by = curve.HIGH/curve.NPOINTS),
                  y=unlist(CWOSL.SAR.Data@DATA[curve.ID[1]]))


##transform values

curve.transformed <- CW2pHMi(curve)

##plot curve
plot(curve.transformed$x, curve.transformed$y.t, log = "x")


##(3) - produce Fig. 4 from Bos & Wallinga (2012)

##load data
data(ExampleData.CW_OSL_Curve, envir = environment())
values <- CW_Curve.BosWallinga2012

##open plot area
plot(NA, NA,
     xlim=c(0.001,10),
     ylim=c(0,8000),
     ylab="pseudo OSL (cts/0.01 s)",
     xlab="t [s]",
     log="x",
```

```
     main="Fig. 4 - Bos & Wallinga (2012)")

values.t<-CW2pLMi(values, P=1/20)
lines(values[1:length(values.t[,1]),1],CW2pLMi(values, P=1/20)[,2],
      col="red" ,lwd=1.3)
text(0.03,4500,"LM", col="red" ,cex=.8)

values.t<-CW2pHMi(values, delta=40)
lines(values[1:length(values.t[,1]),1],CW2pHMi(values, delta=40)[,2],
      col="black", lwd=1.3)
text(0.005,3000,"HM", cex=.8)

values.t<-CW2pPMi(values, P=1/10)
lines(values[1:length(values.t[,1]),1],CW2pPMi(values, P=1/10)[,2],
      col="blue", lwd=1.3)
text(0.5,6500,"PM", col="blue" ,cex=.8)
```

CW2pLM *Transform a CW-OSL curve into a pLM-OSL curve*

Description

Transforms a conventionally measured continuous-wave (CW) curve into a pseudo linearly modulated (pLM) curve using the equations given in Bulur (2000).

Usage

```
CW2pLM(values)
```

Arguments

values	RLum.Data.Curve or data.frame (**required**): RLum.Data.Curve data object. Alternatively, a data.frame of the measured curve data of type stimulation time (t) (values[,1]) and measured counts (cts) (values[,2]) can be provided.

Details

According to Bulur (2000) the curve data are transformed by introducing two new parameters P (stimulation period) and u (transformed time):

$$P = 2 * max(t)$$

$$u = \sqrt{(2 * t * P)}$$

The new count values are then calculated by

$$ctsNEW = cts(u/P)$$

and the returned data.frame is produced by: data.frame(u,ctsNEW)

Value

The function returns the same data type as the input data type with the transformed curve values.

data.frame	generic R data structure
RLum.Data.Curve	package RLum object

Function version

0.4 (2014-04-13 14:29:18)

Note

The transformation is recommended for curves recorded with a channel resolution of at least 0.05 s/channel.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

References

Bulur, E., 2000. A simple transformation for converting CW-OSL curves to LM-OSL curves. Radiation Measurements, 32, 141-145.

Further Reading

Bulur, E., 1996. An Alternative Technique For Optically Stimulated Luminescence (OSL) Experiment. Radiation Measurements, 26, 701-709.

See Also

CW2pHMi, CW2pLMi, CW2pPMi, fit_LMCurve, lm, RLum.Data.Curve

The output of the function can be further used for LM-OSL fitting: CW2pLMi, CW2pHMi, CW2pPMi, fit_LMCurve, RLum.Data.Curve, plot_RLum

Examples

```
##read curve from CWOSL.SAR.Data transform curve and plot values
data(ExampleData.BINfileData, envir = environment())

##read id for the 1st OSL curve
id.OSL <- CWOSL.SAR.Data@METADATA[CWOSL.SAR.Data@METADATA[,"LTYPE"] == "OSL","ID"]

##produce x and y (time and count data for the data set)
x<-seq(CWOSL.SAR.Data@METADATA[id.OSL[1],"HIGH"]/CWOSL.SAR.Data@METADATA[id.OSL[1],"NPOINTS"],
       CWOSL.SAR.Data@METADATA[id.OSL[1],"HIGH"],
     by = CWOSL.SAR.Data@METADATA[id.OSL[1],"HIGH"]/CWOSL.SAR.Data@METADATA[id.OSL[1],"NPOINTS"])
y <- unlist(CWOSL.SAR.Data@DATA[id.OSL[1]])
```

```
values <- data.frame(x,y)

##transform values
values.transformed <- CW2pLM(values)

##plot
plot(values.transformed)
```

CW2pLMi — *Transform a CW-OSL curve into a pLM-OSL curve via interpolation under linear modulation conditions*

Description

Transforms a conventionally measured continuous-wave (CW) OSL-curve into a pseudo linearly modulated (pLM) curve under linear modulation conditions using the interpolation procedure described by Bos & Wallinga (2012).

Usage

```
CW2pLMi(values, P)
```

Arguments

values	RLum.Data.Curve or data.frame (**required**): RLum.Data.Curve or data.frame with measured curve data of type stimulation time (t) (values[,1]) and measured counts (cts) (values[,2])
P	vector (optional): stimulation time in seconds. If no value is given the optimal value is estimated automatically (see details). Greater values of P produce more points in the rising tail of the curve.

Details

The complete procedure of the transformation is given in Bos & Wallinga (2012). The input data.frame consists of two columns: time (t) and count values (CW(t))

Nomenclature

P = stimulation time (s)
1/P = stimulation rate (1/s)

Internal transformation steps

(1) log(CW-OSL) values (2) Calculate t' which is the transformed time:

$$t' = 1/2 * 1/P * t^2$$

(3) Interpolate CW(t'), i.e. use the log(CW(t)) to obtain the count values for the transformed time (t'). Values beyond `min(t)` and `max(t)` produce `NA` values.

(4) Select all values for t' < `min(t)`, i.e. values beyond the time resolution of t. Select the first two values of the transformed data set which contain no `NA` values and use these values for a linear fit using `lm`.

(5) Extrapolate values for t' < `min(t)` based on the previously obtained fit parameters.

(6) Transform values using

$$pLM(t) = t/P * CW(t')$$

(7) Combine values and truncate all values for t' > `max(t)`

The number of values for t' < `min(t)` depends on the stimulation period (P) and therefore on the stimulation rate 1/P. To avoid the production of too many artificial data at the raising tail of the determined pLM curves it is recommended to use the automatic estimation routine for `P`, i.e. provide no own value for `P`.

Value

The function returns the same data type as the input data type with the transformed curve values.

`RLum.Data.Curve`

package `RLum` object with two additional info elements:

$CW2pLMi.x.t	: transformed time values
$CW2pLMi.method	: used method for the production of the new data points

Function version

0.3 (2014-04-13 14:29:25)

Note

According to Bos & Wallinga (2012) the number of extrapolated points should be limited to avoid artificial intensity data. If `P` is provided manually and more than two points are extrapolated, a warning message is returned.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany)

Based on comments and suggestions from:

Adrie J.J. Bos, Delft University of Technology, The Netherlands

R Luminescence Package Team

References

Bos, A.J.J. & Wallinga, J., 2012. How to visualize quartz OSL signal components. Radiation Measurements, 47, 752-758.

Further Reading

Bulur, E., 1996. An Alternative Technique For Optically Stimulated Luminescence (OSL) Experiment. Radiation Measurements, 26, 701-709.

Bulur, E., 2000. A simple transformation for converting CW-OSL curves to LM-OSL curves. Radiation Measurements, 32, 141-145.

See Also

CW2pLM, CW2pHMi, CW2pPMi, fit_LMCurve, RLum.Data.Curve

Examples

```
##(1)
##load CW-OSL curve data
data(ExampleData.CW_OSL_Curve, envir = environment())

##transform values
values.transformed <- CW2pLMi(ExampleData.CW_OSL_Curve)

##plot
plot(values.transformed$x, values.transformed$y.t, log = "x")

##(2) - produce Fig. 4 from Bos & Wallinga (2012)
##load data
data(ExampleData.CW_OSL_Curve, envir = environment())
values <- CW_Curve.BosWallinga2012

##open plot area
plot(NA, NA,
     xlim = c(0.001,10),
     ylim = c(0,8000),
     ylab = "pseudo OSL (cts/0.01 s)",
     xlab = "t [s]",
     log = "x",
     main = "Fig. 4 - Bos & Wallinga (2012)")


values.t <- CW2pLMi(values, P = 1/20)
lines(values[1:length(values.t[,1]),1],CW2pLMi(values, P = 1/20)[,2],
      col = "red", lwd = 1.3)
```

```
text(0.03,4500,"LM", col = "red", cex = .8)

values.t <- CW2pHMi(values, delta = 40)
lines(values[1:length(values.t[,1]),1],CW2pHMi(values, delta = 40)[,2],
      col = "black", lwd = 1.3)
text(0.005,3000,"HM", cex =.8)

values.t <- CW2pPMi(values, P = 1/10)
lines(values[1:length(values.t[,1]),1], CW2pPMi(values, P = 1/10)[,2],
      col = "blue", lwd = 1.3)
text(0.5,6500,"PM", col = "blue", cex = .8)
```

CW2pPMi *Transform a CW-OSL curve into a pPM-OSL curve via interpolation under parabolic modulation conditions*

Description

Transforms a conventionally measured continuous-wave (CW) OSL-curve into a pseudo parabolic modulated (pPM) curve under parabolic modulation conditions using the interpolation procedure described by Bos & Wallinga (2012).

Usage

```
CW2pPMi(values, P)
```

Arguments

values	RLum.Data.Curve or data.frame (**required**): RLum.Data.Curve or data.frame with measured curve data of type stimulation time (t) (values[,1]) and measured counts (cts) (values[,2])
P	vector (optional): stimulation period in seconds. If no value is given, the optimal value is estimated automatically (see details). Greater values of P produce more points in the rising tail of the curve.

Details

The complete procedure of the transformation is given in Bos & Wallinga (2012). The input data.frame consists of two columns: time (t) and count values (CW(t))

Nomenclature

P = stimulation time (s)
1/P = stimulation rate (1/s)

Internal transformation steps

(1) log(CW-OSL) values

(2) Calculate t' which is the transformed time:

$$t' = (1/3) * (1/P^2)t^3$$

(3) Interpolate CW(t'), i.e. use the log(CW(t)) to obtain the count values for the transformed time (t'). Values beyond `min(t)` and `max(t)` produce `NA` values.

(4) Select all values for t' < `min(t)`, i.e. values beyond the time resolution of t. Select the first two values of the transformed data set which contain no `NA` values and use these values for a linear fit using `lm`.

(5) Extrapolate values for t' < `min(t)` based on the previously obtained fit parameters. The extrapolation is limited to two values. Other values at the beginning of the transformed curve are set to 0.

(6) Transform values using

$$pLM(t) = t^2/P^2 * CW(t')$$

(7) Combine all values and truncate all values for t' > `max(t)`

The number of values for t' < `min(t)` depends on the stimulation period `P`. To avoid the production of too many artificial data at the raising tail of the determined pPM curve, it is recommended to use the automatic estimation routine for `P`, i.e. provide no value for `P`.

Value

The function returns the same data type as the input data type with the transformed curve values.

`RLum.Data.Curve`
package `RLum` `object` with two additional info elements:

$CW2pPMi.x.t	: transformed time values
$CW2pPMi.method	: used method for the production of the new data points

`data.frame` with four columns:

$x	: time
$y.t	: transformed count values
$x.t	: transformed time values
$method	: used method for the production of the new data points

Function version

0.2 (2014-04-13 14:29:35)

Note

According to Bos & Wallinga (2012), the number of extrapolated points should be limited to avoid artificial intensity data. If P is provided manually, not more than two points are extrapolated.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany)

Based on comments and suggestions from:
Adrie J.J. Bos, Delft University of Technology, The Netherlands

R Luminescence Package Team

References

Bos, A.J.J. & Wallinga, J., 2012. How to visualize quartz OSL signal components. Radiation Measurements, 47, 752-758.

Further Reading

Bulur, E., 1996. An Alternative Technique For Optically Stimulated Luminescence (OSL) Experiment. Radiation Measurements, 26, 701-709.

Bulur, E., 2000. A simple transformation for converting CW-OSL curves to LM-OSL curves. Radiation Measurements, 32, 141-145.

See Also

CW2pLM, CW2pLMi, CW2pHMi, fit_LMCurve, RLum.Data.Curve

Examples

```
##(1)
##load CW-OSL curve data
data(ExampleData.CW_OSL_Curve, envir = environment())

##transform values
values.transformed <- CW2pPMi(ExampleData.CW_OSL_Curve)

##plot
plot(values.transformed$x,values.transformed$y.t, log = "x")

##(2) - produce Fig. 4 from Bos & Wallinga (2012)

##load data
data(ExampleData.CW_OSL_Curve, envir = environment())
```

```
values <- CW_Curve.BosWallinga2012

##open plot area
plot(NA, NA,
     xlim = c(0.001,10),
     ylim = c(0,8000),
     ylab = "pseudo OSL (cts/0.01 s)",
     xlab = "t [s]",
     log = "x",
     main = "Fig. 4 - Bos & Wallinga (2012)")

values.t <- CW2pLMi(values, P = 1/20)
lines(values[1:length(values.t[,1]),1],CW2pLMi(values, P = 1/20)[,2],
      col = "red",lwd = 1.3)
text(0.03,4500,"LM", col = "red", cex = .8)

values.t <- CW2pHMi(values, delta = 40)
lines(values[1:length(values.t[,1]),1], CW2pHMi(values, delta = 40)[,2],
      col = "black", lwd = 1.3)
text(0.005,3000,"HM", cex = .8)

values.t <- CW2pPMi(values, P = 1/10)
lines(values[1:length(values.t[,1]),1], CW2pPMi(values, P = 1/10)[,2],
      col = "blue", lwd = 1.3)
text(0.5,6500,"PM", col = "blue", cex = .8)
```

`ExampleData.BINfileData`

Example data from a SAR OSL and SAR TL measurement for the package Luminescence

---

Description

Example data from a SAR OSL and TL measurement for package Luminescence directly extracted from a Risoe BIN-file and provided in an object of type Risoe.BINfileData-class

Usage

```
ExampleData.BINfileData
```

Format

`CWOSL.SAR.Data`: SAR OSL measurement data

`TL.SAR.Data`: SAR TL measurement data

Each class object contains two slots: (a) `METADATA` is a data.frame with all metadata stored in the BIN file of the measurements and (b) `DATA` contains a list of vectors of the measured data (usually count values).

Version

0.1

Source

CWOSL.SAR.Data

Lab:	Luminescence Laboratory Bayreuth
Lab-Code:	BT607
Location:	Saxony/Germany
Material:	Middle grain quartz measured on aluminum cups on a Risoe TL/OSL DA-15 reader
Reference:	unpublished

TL.SAR.Data

Lab:	Luminescence Laboratory of Cologne
Lab-Code:	LP1_5
Location:	Spain
Material:	Flint
Setup:	Risoe TL/OSL DA-20 reader (Filter: Semrock Brightline, HC475/50, N2, unpolished steel discs)
Reference:	unpublished
Remarks:	dataset limited to one position

References

CWOSL.SAR.Data: unpublished data

TL.SAR.Data: unpublished data

Examples

```
##show first 5 elements of the METADATA and DATA elements in the terminal
data(ExampleData.BINfileData, envir = environment())
CWOSL.SAR.Data@METADATA[1:5,]
CWOSL.SAR.Data@DATA[1:5]
```

ExampleData.CW_OSL_Curve

Example CW-OSL curve data for the package Luminescence

Description

`data.frame` containing CW-OSL curve data (time, counts)

Usage

```
data(ExampleData.CW_OSL_Curve)
```

Format

Data frame with 1000 observations on the following 2 variables:

`x` a numeric vector, time

`y` a numeric vector, counts

Details

see source

Source

ExampleData.CW_OSL_Curve

Lab:	Luminescence Laboratory Bayreuth
Lab-Code:	BT607
Location:	Saxony/Germany
Material:	Middle grain quartz measured on aluminum cups on a Risoe TL/OSL DA-15 reader.
Reference:	unpublished data

CW_Curve.BosWallinga2012

Lab:	Netherlands Centre for Luminescence Dating (NCL)
Lab-Code:	NCL-2108077
Location:	Guadalentin Basin, Spain
Material:	Coarse grain quartz
Reference:	Bos & Wallinga (2012) and Baartman et al. (2011)

References

Baartman, J.E.M., Veldkamp, A., Schoorl, J.M., Wallinga, J., Cammeraat, L.H., 2011. Unravelling Late Pleistocene and Holocene landscape dynamics: The Upper Guadalentin Basin, SE Spain. Geomorphology, 125, 172-185.

Bos, A.J.J. & Wallinga, J., 2012. How to visualize quartz OSL signal components. Radiation Measurements, 47, 752-758.

Examples

```
data(ExampleData.CW_OSL_Curve, envir = environment())
plot(ExampleData.CW_OSL_Curve)
```

ExampleData.DeValues *Example De data for the package Luminescence*

Description

25 equivalent dose (De) values measured for a fine grain quartz sample from a loess section in Rottewitz (Saxony/Germany).

Usage

```
ExampleData.DeValues
```

Format

A data.frame with two columns:

x a numeric vector, De

y a numeric vector, De error

Source

Lab:	Luminescence Laboratory Bayreuth
Lab-Code:	BT998
Location:	Rottewitz (Saxony/Germany)
Material:	Fine grain quartz measured on aluminum discs on a Risoe TL/OSL DA-15 reader
Units:	Values are given in seconds
Dose Rate:	Dose rate of the beta-source at measurement ca. 0.0438 Gy/s +/- 0.0019 Gy/s

References

unpublished data

Examples

```
##(1) plot values as histogram
data(ExampleData.DeValues, envir = environment())
plot_Histogram(ExampleData.DeValues, xlab = "De [s]")

##(2) plot value as histogram (with Second to Gray convertion)
data(ExampleData.DeValues, envir = environment())

De.values <- Second2Gray(ExampleData.DeValues,
                         dose_rate = c(0.0438, 0.0019),
                         method = "gaussian")
```

```
plot_Histogram(De.values, xlab = "De [Gy]")
```

ExampleData.FittingLM *Example data for fit_LMCurve() in the package Luminescence*

Description

Lineraly modulated (LM) measurement data from a quartz sample from Norway including background measurement. Measurements carried out in th luminescence laboratory at the University of Bayreuth.

Usage

```
ExampleData.FittingLM
```

Format

Two objects (data.frames) with two columns (time and counts).

Source

Lab:	Luminescence Laboratory Bayreuth
Lab-Code:	BT900
Location:	Norway
Material:	Beach deposit, coarse grain quartz measured on aluminum discs on a Risoe TL/OSL DA-15 reader

References

Fuchs, M., Kreutzer, S., Fischer, M., Sauer, D., Soerensen, R., 2012. OSL and IRSL dating of raised beach sand deposits along the southeastern coast of Norway. Quaternary Geochronology, 10, 195-200.

Examples

```
##show LM data
data(ExampleData.FittingLM, envir = environment())
plot(values.curve,log="x")
```

ExampleData.LxTxData *Example Lx/Tx data from CW-OSL SAR measurement*

Description

LxTx data from a SAR measurement for the package Luminescence.

Usage

```
ExampleData.LxTxData
```

Format

A `data.frame` with 4 columns (Dose, LxTx, LxTx.Error, TnTx).

Source

Lab:	Luminescence Laboratory Bayreuth
Lab-Code:	BT607
Location:	Ostrau (Saxony-Anhalt/Germany)
Material:	Middle grain quartz measured on a Risoe TL/OSL DA-15 reader.

References

unpublished data

Examples

```
##plot Lx/Tx data vs dose [s]
data(ExampleData.LxTxData, envir = environment())
plot(LxTxData$Dose,LxTxData$LxTx)
```

ExampleData.LxTxOSLData *Example Lx and Tx curve data from an artificial OSL measurement*

Description

Lx and Tx data of continous wave (CW-) OSL signal curves.

Usage

```
ExampleData.LxTxOSLData.RData
```

Format

Two `data.frames` containing time and count values.

Source

Arbitrary OSL measurement.

References

unpublished data

Examples

```
##load data
data(ExampleData.LxTxOSLData, envir = environment())

##plot data
plot(Lx.data)
plot(Tx.data)
```

`ExampleData.RLum.Analysis`
Example data as `RLum.Analysis` *objects*

Description

Collection of different `RLum.Analysis` objects for protocol analysis.

Usage

```
ExampleData.RLum.Analysis
```

Format

`IRSAR.RF.Data`: IRSAR.RF.Data on coarse grain feldspar

Each object contains data needed for the given protocol analysis.

Version

0.1

Source

IRSAR.RF.Data

These data have been kindly provided by Tobias Lauer and Matthias Krbetschek.

Lab:	Luminescence Laboratory TU Bergakademie Freiberg
Lab-Code:	ZEU/SA1
Location:	Zeuchfeld (Zeuchfeld Sandur; Saxony-Anhalt/Germany)
Material:	K-feldspar (130-200 μm)
Reference:	Kreutzer et al. (2012)

References

IRSAR.RF.Data

Kreutzer, S., Lauer, T., Meszner, S., Krbetschek, M.R., Faust, D., Fuchs, M., 2012. Chronology of the Quaternary profile Zeuchfeld in Saxony-Anhalt / Germany - a preliminary luminescence dating study. Zeitschrift fuer Geomorphologie fast track, 1-21. doi: 10.1127/0372-8854/2012/S-00112

Examples

```
##load data
data(ExampleData.RLum.Analysis, envir = environment())

##plot data
plot_RLum(IRSAR.RF.Data)
```

ExampleData.XSYG *Example data for a SAR OSL measurement and a TL spectrum using a lexsyg reader*

Description

Example data from a SAR OSL measurement and a TL spectrum for package Luminescence imported from a Freiberg Instruments XSYG file using the function `readXSYG2R`.

Usage

```
ExampleData.XSYG
```

Format

`OSL.SARMeasurement`: SAR OSL measurement data

The data contain two elements: (a) `$Sequence.Header` is a data.frame with metadata from the measurement,(b) `Sequence.Object` contains an `RLum.Analysis` object for further analysis.

`TL.Spectrum`: TL spectrum data

RLum.Data.Spectrum object for further analysis. The spectrum was cleaned from cosmic-rays using the function, apply_CosmicRayRemoval. Note that no quantum efficiency calibration was performed.

Version

0.1

Source

OSL.SARMeasurement

Lab:	Luminescence Laboratory Giessen
Lab-Code:	no code
Location:	not specified
Material:	Coarse grain quartz on steel cups on lexsyg research reader
Reference:	unpublished

TL.Spectrum

Lab:	Luminescence Laboratory Giessen
Lab-Code:	BT753
Location:	Dolni Vestonice/Czech Republic
Material:	Fine grain polymineral on steel cups on lexsyg rearch reader
Reference:	Fuchs et al., 2013
Spectrum:	Integration time 19 s, channel time 20 s
Heating:	1 K/s, up to 500 deg. C

References

Unpublished data measured to serve as example data for that package. Location origin of sample BT753 is given here:

Fuchs, M., Kreutzer, S., Rousseau, D.D., Antoine, P., Hatte, C., Lagroix, F., Moine, O., Gauthier, C., Svoboda, J., Lisa, L., 2013. The loess sequence of Dolni Vestonice, Czech Republic: A new OSL-based chronology of the Last Climatic Cycle. Boreas, 42, 664–677.

See Also

readXSYG2R, RLum.Analysis, RLum.Data.Spectrum, plot_RLum, plot_RLum.Analysis, plot_RLum.Data.Spectrum

Examples

```
##show data
data(ExampleData.XSYG, envir = environment())
```

```
## ========================================
##(1) OSL.SARMeasurement
OSL.SARMeasurement

##show $Sequence.Object
OSL.SARMeasurement$Sequence.Object

##grep OSL curves and plot the first curve
OSLcurve <- get_RLum.Analysis(OSL.SARMeasurement$Sequence.Object,
recordType="OSL")[[1]]
plot_RLum(OSLcurve)

## ========================================
##(2) TL.Spectrum
TL.Spectrum

##plot simple spectrum (2D)
plot_RLum.Data.Spectrum(TL.Spectrum,
                        plot.type="contour",
                        xlim = c(310,750),
                        ylim = c(0,300),
                        bin.rows=10,
                        bin.cols = 1)

##plot 3d spectrum (uncomment for usage)
# plot_RLum.Data.Spectrum(TL.Spectrum, plot.type="persp",
# xlim = c(310,750), ylim = c(0,300), bin.rows=10,
# bin.cols = 1)
```

`fit_CWCurve` *Nonlinear Least Squares Fit for CW-OSL curves [beta version]*

Description

The function determines the weighted least-squares estimates of the component parameters of a CW-OSL signal for a given maximum number of components and returns various component parameters. The fitting procedure uses the `nls` function with the `port` algorithm.

Usage

```
fit_CWCurve(values, n.components.max, fit.failure_threshold = 3,
    fit.trace = FALSE, fit.calcError = FALSE, LED.power = 36,
    LED.wavelength = 470, log = "", cex.global = 0.6, main = "CW-OSL Curve Fit",
    sample_code = "Default", ylab, xlab, output.path, output.terminal = TRUE,
    output.terminalAdvanced = TRUE, output.plot = TRUE)
```

Arguments

values	RLum.Data.Curve or data.frame (**required**): x, y data of measured values (time and counts). See examples.
n.components.max	vector (optional): maximum number of components that are to be used for fitting. The upper limit is 7.
fit.failure_threshold	vector (with default): limits the failed fitting attempts.
fit.trace	logical (with default): traces the fitting process on the terminal.
fit.calcError	logical (with default): calculate 1-sigma error range of components using confint
LED.power	numeric (with default): LED power (max.) used for intensity ramping in mW/cm^2. **Note:** The value is used for the calculation of the absolute photoionisation cross section.
LED.wavelength	numeric (with default): LED wavelength used for stimulation in nm. **Note:** The value is used for the calculation of the absolute photoionisation cross section.
log	character (optional): option for log-scaled axis, works as in plot
cex.global	numeric (with default): global scaling factor.
main	character (with default): header for plot output.
sample_code	character (optional): sample code used for the plot and the optional output table (mtext).
ylab	character (with default): alternative y-axis labelling
xlab	character (with default): alternative x-axis labelling
output.path	character (optional): output path for table output containing the results of the fit. The file name is set automatically. If the file already exists in the directory, the values are appended.
output.terminal	logical (with default): terminal ouput with fitting results.
output.terminalAdvanced	logical (with default): enhanced terminal output. Requires output.terminal = TRUE. If output.terminal = FALSE no advanced output is possible.
output.plot	logical (with default): returns a plot of the fitted curves.

Details

Fitting function

The function for the CW-OSL fitting has the general form:

$$y = I0_1 * \lambda_1 * exp(-\lambda_1 * x) + , \ldots , + I0_i * \lambda_i * exp(-\lambda_i * x)$$

where $1 < i < 8$

and λ is the decay constant and $N0$ the intial number of trapped electrons.
(for the used equation cf. Boetter-Jensen et al., 2003)

Start values

Start values are estimated automatically by fitting a linear function to the logarithmized input data set. Currently, there is no option to manually provide start parameters.

Goodness of fit

The goodness of the fit is given as pseudoR^2 value (pseudo coefficient of determination). According to Lave (1970), the value is calculated as:

$$pseudoR^2 = 1 - RSS/TSS$$

where $RSS = Residual\tilde{}Sum\tilde{}of\tilde{}Squares$
and $TSS = Total\tilde{}Sum\tilde{}of\tilde{}Squares$

Error of fitted component parameters

The 1-sigma error for the components is calculated using the function `confint`. Due to considerable calculation time, this option is deactived by default. In addition, the error for the components can be estimated by using internal R functions like `summary`. See the `nls` help page for more information.

For details on the nonlinear regression in R, see Ritz & Streibig (2008).

Value

`plot` (optional) the fitted CW-OSL curves are returned as plot.

`table` (optional) an output table (*.csv) with parameters of the fitted components is provided if the `output.path` is set.

`RLum.Results` object

beside the plot and table output options, an `RLum.Results` object is returned.

`fit`: an `nls` object (`$fit`) for which generic R functions are provided, e.g. summary, confint, profile. For more details, see nls.

`output.table`: a data.frame containing the summarized parameters including the error
`component.contribution.matrix`: matrix containing the values for the component to sum contribution plot (`$component.contribution.matrix`).

Matrix structure:
Column 1 and 2: time and `rev(time)` values
Additional columns are used for the components, two for each component, containing I0 and n0. The last columns `cont.` provide information on the relative component contribution for each time interval including the row sum for this values.

Function version

0.4.3 (2014-04-13 14:29:43)

Note

Beta version - This function has not been properly tested yet and should therefore not be used for publication purposes!

The pseudo-R^2 may not be the best parameter to describe the goodness of the fit. The trade off between the `n.components` and the pseudo-R^2 value is currently not considered.

The function **does not** ensure that the fitting procedure has reached a global minimum rather than a local minimum!

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

References

Boetter-Jensen, L., McKeever, S.W.S., Wintle, A.G., 2003. Optically Stimulated Luminescence Dosimetry. Elsevier Science B.V.

Lave, C.A.T., 1970. The Demand for Urban Mass Transportation. The Review of Economics and Statistics, 52 (3), 320-323.

Ritz, C. & Streibig, J.C., 2008. Nonlinear Regression with R. In: R. Gentleman, K. Hornik, G. Parmigiani, eds., Springer, p. 150.

See Also

`fit_LMCurve`, `plot`,`nls`, `RLum.Data.Curve`, `RLum.Results`, `get_RLum.Results`

Examples

```
##load data
data(ExampleData.CW_OSL_Curve, envir = environment())

##fit data
fit <- fit_CWCurve(values = ExampleData.CW_OSL_Curve,
                   main = "CW Curve Fit",
                   n.components.max = 4,
                   log = "x")
```

fit_LMCurve *Nonlinear Least Squares Fit for LM-OSL curves*

Description

The function determines weighted nonlinear least-squares estimates of the component parameters of an LM-OSL curve (Bulur 1996) for a given number of components and returns various component parameters. The fitting procedure uses the function nls with the port algorithm.

Usage

```
fit_LMCurve(values, values.bg, n.components = 3, start_values,
    input.dataType = "LM", sample_code = "", sample_ID = "",
    LED.power = 36, LED.wavelength = 470, cex.global = 0.8, fit.trace = FALSE,
    fit.advanced = FALSE, fit.calcError = FALSE, bg.subtraction = "polynomial",
    output.path, output.terminal = TRUE, output.terminaladvanced = TRUE,
    output.plot = TRUE, output.plotBG = FALSE, ...)
```

Arguments

values	RLum.Data.Curve or data.frame (**required**): x,y data of measured values (time and counts). See examples.
values.bg	RLum.Data.Curve or data.frame (optional): x,y data of measured values (time and counts) for background subtraction.
n.components	integer (with default): fixed number of components that are to be recognised during fitting (min = 1, max = 7).
start_values	data.frame (optional): start parameters for lm and xm data for the fit. If no start values are given, an automatic start value estimation is attempted (see details).
input.dataType	character (with default): alter the plot output depending on the input data: "LM" or "pLM" (pseudo-LM). See: CW2pLM
sample_code	character (optional): sample code used for the plot and the optional output table (mtext).
sample_ID	character (optional): additional identifier used as column header for the table output.
LED.power	numeric (with default): LED power (max.) used for intensity ramping in mW/cm^2. **Note:** This value is used for the calculation of the absolute photoionisation cross section.
LED.wavelength	numeric (with default): LED wavelength in nm used for stimulation. **Note:** This value is used for the calculation of the absolute photoionisation cross section.
cex.global	numeric (with default): global scaling factor.
fit.trace	logical (with default): traces the fitting process on the terminal.
fit.advanced	logical (with default): enables advanced fitting attempt for automatic start parameter recognition. Works only if no start parameters are provided. **Note:** It may take a while.

`fit.calcError`	logical (with default): calculate 1-sigma error range of components using confint.
`bg.subtraction`	character (with default): specifies method for background subtraction (`polynomial`, `linear`, `channel`, see Details). **Note:** requires input for `values.bg`.
`output.path`	character (optional): output path for table output containing the results of the fit. The file name is set automatically. If the file already exists in the directory, the values are appended.
`output.terminal`	logical (with default): terminal output with fitting results.
`output.terminaladvanced`	logical (with default): enhanced terminal output. Requires `output.terminal = TRUE`. If `output.terminal = FALSE` no advanced output is possible.
`output.plot`	logical (with default): returns a plot of the fitted curves.
`output.plotBG`	logical (with default): returns a plot of the background values with the fit used for the background subtraction.
`...`	Further arguments that may be passed to the plot output, e.g. `xlab`, `xlab`, `main`, `log`.

Details

Fitting function

The function for the fitting has the general form:

$$y = (exp(0.5)*Im_1*x/xm_1)*exp(-x^2/(2*xm_1^2))+,\ldots,+exp(0.5)*Im_i*x/xm_i)*exp(-x^2/(2*xm_i^2))$$

where $1 < i < 8$
This function and the equations for the conversion to b (detrapping probability) and n0 (proportional to initially trapped charge) have been taken from Kitis et al. (2008):

$$xm_i = \sqrt{max(t)/b_i}$$

$$Im_i = exp(-0.5)n0/xm_i$$

Background subtraction

Three methods for background subtraction are provided for a given background signal (`values.bg`).
`polynomial`: default method. A polynomial function is fitted using glm and the resulting function is used for background subtraction:

$$y = a * x^4 + b * x^3 + c * x^2 + d * x + e$$

`linear`: a linear function is fitted using glm and the resulting function is used for background subtraction:

$$y = a * x + b$$

`channel`: the measured background signal is subtracted channelwise from the measured signal.

Start values

The choice of the initial parameters for the `nls`-fitting is a crucial point and the fitting procedure may mainly fail due to ill chosen start parameters. Here, three options are provided:

(a) If no start values (`start_values`) are provided by the user, a cheap guess is made by using the detrapping values found by Jain et al. (2003) for quartz for a maximum of 7 components. Based on these values, the pseudo start parameters xm and Im are recalculated for the given data set. In all cases, the fitting starts with the ultra-fast component and (depending on `n.components`) steps through the following values. If no fit could be achieved, an error plot (for `output.plot = TRUE`) with the pseudo curve (based on the pseudo start parameters) is provided. This may give the opportunity to identify appropriate start parameters visually.

(b) If start values are provided, the function works like a simple `nls` fitting approach.

(c) If no start parameters are provided and the option `fit.advanced = TRUE` is chosen, an advanced start paramter estimation is applied using a stochastical attempt. Therefore, the recalculated start parameters **(a)** are used to construct a normal distribution. The start parameters are then sampled randomly from this distribution. A maximum of 100 attempts will be made. **Note:** This process may be time consuming.

Goodness of fit

The goodness of the fit is given by a pseudoR^2 value (pseudo coefficient of determination). According to Lave (1970), the value is calculated as:

$$pseudoR^2 = 1 - RSS/TSS$$

where $RSS = Residual\tilde{}Sum\tilde{}of\tilde{}Squares$
and $TSS = Total\tilde{}Sum\tilde{}of\tilde{}Squares$

Error of fitted component parameters

The 1-sigma error for the components is calculated using the function confint. Due to considerable calculation time, this option is deactived by default. In addition, the error for the components can be estimated by using internal R functions like summary. See the nls help page for more information.
For more details on the nonlinear regression in R, see Ritz & Streibig (2008).

Value

`plot`	(optional) various types of plots are returned. For details see above.
`table`	(optional) an output table (*.csv) with the fitted components is provided if the `output.path` is set.
`list object`	beside the plot and table output, a list is returned. The list contains: (a) an `nls` object (`$fit`) for which generic R functions are provided, e.g. summary, confint, profile. For more details, see nls.

(b) a data.frame containing the summarised parameters including the error (`$output.table`).
(c) a matrix containing the values for the component to sum contribution plot (`$component.contribution.matrix`).

Matrix structure:
Column 1 and 2: time and `rev(time)` values
Additional columns are used for the components, two for each component, containing I0 and n0. The last columns `cont.` provide information on the relative component contribution for each time interval including the row sum for this values.

Function version

0.2.15 (2014-04-13 14:29:53)

Note

The pseudo-R^2 may not be the best parameter to describe the goodness of the fit. The trade off between the `n.components` and the pseudo-R^2 value currently remains unconsidered.

The function **does not** ensure that the fitting procedure has reached a global minimum rather than a local minimum! In any case of doubt, the use of manual start values is highly recommended.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

References

Bulur, E., 1996. An Alternative Technique For Optically Stimulated Luminescence (OSL) Experiment. Radiation Measurements, 26, 5, 701-709.

Jain, M., Murray, A.S., Boetter-Jensen, L., 2003. Characterisation of blue-light stimulated luminescence components in different quartz samples: implications for dose measurement. Radiation Measurements, 37 (4-5), 441-449.

Kitis, G. & Pagonis, V., 2008. Computerized curve deconvolution analysis for LM-OSL. Radiation Measurements, 43, 737-741.

Lave, C.A.T., 1970. The Demand for Urban Mass Transportation. The Review of Economics and Statistics, 52 (3), 320-323.

Ritz, C. & Streibig, J.C., 2008. Nonlinear Regression with R. R. Gentleman, K. Hornik, & G. Parmigiani, eds., Springer, p. 150.

See Also

`fit_CWCurve`, `plot`, `nls`

Examples

```
##(1) fit LM data without background subtraction
data(ExampleData.FittingLM, envir = environment())
fit_LMCurve(values = values.curve, n.components = 3, log = "x")

##(2) fit LM data with background subtraction and export as JPEG
## -alter file path for your preferred system
##jpeg(file = "~/Desktop/Fit_Output\%03d.jpg", quality = 100,
## height = 3000, width = 3000, res = 300)
data(ExampleData.FittingLM, envir = environment())
fit_LMCurve(values = values.curve, values.bg = values.curveBG,
            n.components = 2, log = "x", output.plotBG = TRUE)
##dev.off()

##(3) fit LM data with manual start parameters
data(ExampleData.FittingLM, envir = environment())
fit_LMCurve(values = values.curve,
            values.bg = values.curveBG,
            n.components = 3,
            log = "x",
            start_values = data.frame(Im = c(170,25,400), xm = c(56,200,1500)))
```

`merge_Risoe.BINfileData`

Merge Risoe.BINfileData objects or Risoe BIN-files

Description

Function allows merging Risoe BIN files or Risoe.BINfileData objects.

Usage

```
merge_Risoe.BINfileData(input.objects, output.file, keep.position.number = FALSE,
    position.number.append.gap = 0)
```

Arguments

`input.objects`	`character` or `Risoe.BINfileData` (**required**): Character vector with path and files names (e.g. `input.objects = c("path/file1.bin", "path/file2.bin")` or `Risoe.BINfileData` objects (e.g. `input.objects = c(object1, object2)`)
`output.file`	`character` (optional): File output path and name. If no value is given, a `Risoe.BINfileData` is returned instead of a file.
`keep.position.number`	`logical` (with default): Allows keeping the original position numbers of the input objects. Otherwise the position numbers a recalculated.

position.number.append.gap
: integer (with default): Set the position number gap between merged BIN-file sets, if the option keep.position.number = FALSE is used. See details for further information.

Details

The function allows mering different measurements to one file or one object.
The record IDs are recalculated for the new object. Other values are kept for each object. The number of input objects is not limited.

position.number.append.gap option

If the option keep.position.number = FALSE is used, the position numbers of the new data set are recalculated by adding the highest position number of the previous data set to the each position number of the next data set. For example: The highest position number is 48, then this number will be added to all other position numbers of the next data set (e.g. 1 + 48 = 49)

However, there might be cases where an additional addend (summand) is needed before the next position starts. Example:

Position number set (A): 1,3,5,7
Position number set (B): 1,3,5,7

With no additional summand the new position numbers would be: 1,3,5,7,8,9,10,11. That might be unwanted. Using the argument position.number.append.gap = 1 it will become: 1,3,5,7,9,11,13,15,17.

Value

Returns a file or a Risoe.BINfileData object.

Function version

0.2 (2014-04-13 14:30:01)

Note

The validity of the output objects is not further checked.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

References

Duller, G., 2007. Analyst.

See Also

Risoe.BINfileData, readBIN2R, writeR2BIN

Examples

```
##merge two objects
data(ExampleData.BINfileData, envir = environment())

object1 <- CWOSL.SAR.Data
object2 <- CWOSL.SAR.Data

object.new <- merge_Risoe.BINfileData(c(object1, object2))
```

---

plot_AbanicoPlot *Function to create an Abanico Plot.*

Description

A plot is produced which allows comprehensive presentation of data precision and its dispersion around a central value as well as illustration of a kernel density estimate of the dose values.

Usage

```
plot_AbanicoPlot(data, na.exclude = TRUE, log.z = TRUE, central.value,
    centrality = "mean.weighted", dispersion = "sd", plot.ratio = 0.75,
    mtext, summary, summary.pos, legend, legend.pos, stats, y.axis = TRUE,
    error.bars = FALSE, polygon.col, bar.col, line, line.col,
    line.label, grid.col, bw = "nrd0", output = FALSE, ...)
```

Arguments

data	data.frame or RLum.Results object (required): for data.frame two columns: De (data[,1]) and De error (data[,2]). To plot several data sets in one plot the data sets must be provided as list, e.g. list(data.1, data.2).
na.exclude	logical (with default): exclude NA values from the data set prior to any further operations.
log.z	logical (with default): Option to display the z-axis in logarithmic scale. Default is TRUE.
central.value	numeric: User-defined central value, primarily used for horizontal centering of the z-axis.
centrality	character (with default): measure of centrality, used for automatically centering the plot and drawing the central line. Can be one out of "mean", "median", "mean.weighted", and "median.weighted". Default is "mean.weighted".

dispersion	character (with default): measure of dispersion, used for drawing the polygon that depicts the dose distribution. One out of "sd" (standard deviation),"2sd" (2 standard deviations) "qr" (quartile range), default is "sd".
plot.ratio	numeric: Relative space, given to the radial versus the cartesian plot part, deault is 0.75.
mtext	character: additional text below the plot title.
summary	character (optional): adds numerical output to the plot. Can be one or more out of: "n" (number of samples), "mean" (mean De value), "mean.weighted" (error-weighted mean), "median" (median of the De values), "sdrel" (relative standard deviation in percent), "sdabs" (absolute standard deviation), "serel" (relative standard error), "seabs" (absolute standard error) and "in.ci" (percent of samples in confidence interval, e.g. 2-sigma).
summary.pos	numeric or character (with default): optional position coordinates or keyword (e.g. "topright") for the statistical summary. Alternatively, the keyword "sub" may be specified to place the summary below the plot header. However, this latter option in only possible if mtext is not used.
legend	character vector (optional): legend content to be added to the plot.
legend.pos	numeric or character (with default): optional position coordinates or keyword (e.g. "topright") for the legend to be plotted.
stats	character: additional labels of statistically important values in the plot. One or more out of the following: "min", "max", "median".
y.axis	logical: Option to hide y-axis labels. Useful for data with small scatter.
error.bars	logical: Option to show De-errors as error bars on De-points. Useful in combination with y.axis = FALSE, bar.col = "none".
polygon.col	character or numeric (with default): colour of the polygon showing the dose dispersion around the central value. To disable the polygon use "none". Default is "grey80".
bar.col	character or numeric (with default): colour of the bar showing the 2-sigma range of the dose error around the central value. To disable the bar use "none". Default is "grey50".
line	numeric: numeric values of the additional lines to be added.
line.col	character or numeric: colour of the additional lines.
line.label	character: labels for the additional lines.
grid.col	character or numeric (with default): colour of the grid lines (originating at [0,0] and strechting to the z-scale). To disable grid lines use "none". Default is "grey".
bw	character (with default): bin-width for KDE, choose a numeric value for manual setting.
output	logical: Optional output of numerical plot parameters. These can be useful to reproduce similar plots. Default is FALSE.
...	Further plot arguments to pass. xlab must be a vector of length 2, specifying the upper and lower x-axes labels.

Details

The Abanico Plot is a combination of the classic Radial Plot (`plot_RadialPlot`) and a kernel density estimate plot (e.g. `plot_KDE`). It allows straightforward visualisation of data precision, error scatter around a user-defined central value and the combined distribution of the values, on the actual scale of the measured data (e.g. seconds, equivalent dose, years). The principle of the plot is shown in Galbraith & Green (1990). The function authors are thankful for the thoughtprovocing figure in this article.
The semi circle (z-axis) of the classic Radial Plot is bent to a straight line here, which actually is the basis for combining this polar (radial) part of the plot with any other cartesian visualisation method (KDE, histogram, PDF and so on). Note that the plot allows dispaying two measures of distribution. One is the 2-sigma bar, which illustrates the spread in value errors, and the other is the polygon, which stretches over both parts of the Abanico Plot (polar and cartesian) and illustrates the actual spread in the values themselfes.
Since the 2-sigma-bar is a polygon, it can be (and is) filled with shaded lines. To change density (lines per inch, default is 15) and angle (default is 45 degrees) of the shading lines, specify these parameters. See `?polygon()` for further help.
The Abanico Plot supports other than the weighted mean as measure of centrality. When it is obvious that the data is not (log-)normally distributed, the mean (weighted or not) cannot be a valid measure of centrality and hence central dose. Accordingly, the median and the weighted median can be chosen as well to represent a proper measure of centrality (e.g. `centrality = "median.weighted"`).
The proportion of the polar part and the cartesian part of the Abanico Plot can be modfied for display reasons (`plot.ratio = 0.75`). By default, the polar part spreads over 75 % and leaves 25 % for the part that shows the KDE graph.

Value

returns a plot object and, optionally, a list with plot calculus data.

Function version

0.1 (2014-06-05 13:42:28)

Author(s)

Michael Dietze, GFZ Potsdam (Germany), Sebastian Kreutzer, JLU Giessen (Germany)
Inspired by a plot introduced by Galbraith & Green (1990)
R Luminescence Package Team

References

Galbraith, R. & Green, P., 1990. Estimating the component ages in a finite mixture. International Journal of Radiation Applications and Instrumentation. Part D. Nuclear Tracks and Radiation Measurements, 17 (3), pp. 197-206.

See Also

`plot_RadialPlot`, `plot_KDE`, `plot_Histogram`

Examples

```
## load example data and recalculate to Gray
data(ExampleData.DeValues, envir = environment())
ExampleData.DeValues <-
  Second2Gray(values = ExampleData.DeValues, dose_rate = c(0.0438,0.0019))

## plot the example data straightforward
plot_AbanicoPlot(data = ExampleData.DeValues)

## now with linear z-scale
plot_AbanicoPlot(data = ExampleData.DeValues,
                 log.z = FALSE)

## now with output of the plot parameters
plot1 <- plot_AbanicoPlot(data = ExampleData.DeValues,
                          output = TRUE)
str(plot1)
plot1$zlim

## now with adjusted z-scale limits
plot_AbanicoPlot(data = ExampleData.DeValues,
                 zlim = c(100, 200))

## now with adjusted x-scale limits
plot_AbanicoPlot(data = ExampleData.DeValues,
                 xlim = c(0, 60))

## now with user-defined plot ratio
plot_AbanicoPlot(data = ExampleData.DeValues,
                 plot.ratio = 0.5)

## now with user-defined central value
plot_AbanicoPlot(data = ExampleData.DeValues,
                 central.value = 120)

## now with weighted median as measure of centrality
plot_AbanicoPlot(data = ExampleData.DeValues,
                 centrality = "median.weighted")

## now with median/quartile range as measure of centrality/dispersion
plot_AbanicoPlot(data = ExampleData.DeValues,
                 centrality = "median",
                 dispersion = "qr")

## now with user-defined green line for MAM3 (i.e. 2936.3)
MAM <- calc_MinDose3(input.data = ExampleData.DeValues,
                     sigmab = 0.3,
                     gamma.xub = 7000,
                     output.plot = FALSE)

MAM <- as.numeric(get_RLum.Results(object = MAM,
                                   data.object = "results")$mindose)
```

```
plot_AbanicoPlot(data = ExampleData.DeValues,
                 xlim = c(0, 50),
                 line = MAM,
                 line.col = "darkgreen",
                 line.label = "MAM3-dose")

## now add lines (e.g. De = 100) completely manually
## 1. infer extra data
extra <- plot_AbanicoPlot(data = ExampleData.DeValues,
                          output = TRUE)

## 2. transform De value to plot coordinates, only use log when
##    log.z = TRUE. Don't mind the cryptic equation too much.
De <- 100
y.De <- (log(De) - extra$data.global$z.central[1]) * extra$polar.box[2]

## 3. create line coordinates (origin - polar margin - cartesian margin)
line.x <- c(0, extra$polar.box[2], extra$cartesian.box[2])
line.y <- c(0, y.De, y.De)

## 4. draw the line
lines(x = line.x, y = line.y, lwd = 2, lty = 4, col = "tomato")

## now create plot with legend, colour, different points and smaller scale
plot_AbanicoPlot(data = ExampleData.DeValues,
                 legend = "Sample 1",
                 col = "tomato4",
                 bar.col = "peachpuff",
                 pch = "R",
                 cex = 0.8)

## now without 2-sigma bar, polygon, grid lines and central value line
plot_AbanicoPlot(data = ExampleData.DeValues,
                 bar.col = "none",
                 polygon.col = "none",
                 grid.col = "none",
                 y.axis = FALSE,
                 lwd = 0)

## now with direct display of De errors, without 2-sigma bar
plot_AbanicoPlot(data = ExampleData.DeValues,
                 bar.col = "none",
                 ylab = "",
                 y.axis = FALSE,
                 error.bars = TRUE)

## now with user-defined axes labels
plot_AbanicoPlot(data = ExampleData.DeValues,
                 xlab = c("Data error [%]",
                          "Data precision"),
                 ylab = "Scatter",
                 zlab = "Equivalent dose [Gy]")
```

```
## now with minimum, maximum and median value indicated
plot_AbanicoPlot(data = ExampleData.DeValues,
                 central.value = 150,
                 stats = c("min", "max", "median"))

## now with a brief statistical summary
plot_AbanicoPlot(data = ExampleData.DeValues,
                 summary = c("n", "in.ci"))

## now with another statistical summary as subheader
plot_AbanicoPlot(data = ExampleData.DeValues,
                 summary = c("mean.weighted", "median"),
                 summary.pos = "sub")

## now the data set is split into sub-groups, one is manipulated
data.1 <- ExampleData.DeValues[1:15,]
data.2 <- ExampleData.DeValues[16:25,] * 1.3

## now a common dataset is created from the two subgroups
data.3 <- list(data.1, data.2)

## now the two data sets are plotted in one plot
plot_AbanicoPlot(data = data.3)

## now with some graphical modification
plot_AbanicoPlot(data = data.3,
                 col = c("steelblue4", "orange4"),
                 bar.col = c("steelblue3", "orange3"),
                 polygon.col = c("steelblue1", "orange1"),
                 pch = c(2, 6),
                 density = c(10, 20),
                 angle = c(30, 50),
                 summary = c("n", "in.ci"))
```

`plot_DRTResults` *Visualise dose recovery test results*

Description

The function provides a standardised plot output for dose recovery test measurements.

Usage

```
plot_DRTResults(values, given.dose, error.range = 10, preheat,
    boxplot = FALSE, mtext, summary, summary.pos, legend, legend.pos,
    na.exclude = FALSE, ...)
```

Arguments

values	RLum.Results or data.frame, (**required**): input values containing at least De and De error. To plot more than one data set in one figure, a list of the individual data sets must be provided (e.g. list(dataset.1, dataset.2)).
given.dose	numeric: given dose from the dose recovery test (in Seconds or Gray, unit has to be the same as from the input values).
error.range	numeric: symmetric error range in percent will be shown as dashed lines in the plot. Set error.range to 0 to void plotting of error ranges.
preheat	numeric: optional vector of preheat temperatures to be used for grouping the De values. If specified, the temperatures are assigned to the x-axis.
boxplot	logical: optionally plot values, that are grouped by preheat temperature as boxplots. Only possible when preheat vector is specified.
mtext	character: additional text below the plot title.
summary	character (optional): adds numerical output to the plot. Can be one or more out of: "n" (number of samples), "mean" (mean De value), "mean.weighted" (error-weighted mean), "median" (median of the De values), "sdrel" (relative standard deviation in percent), "sdabs" (absolute standard deviation), "serel" (relative standard error) and "seabs" (absolute standard error).
summary.pos	numeric or character (with default): optional position coordinates or keyword (e.g. "topright") for the statistical summary. Alternatively, the keyword "sub" may be specified to place the summary below the plot header. However, this latter option in only possible if mtext is not used.
legend	character vector (optional): legend content to be added to the plot.
legend.pos	numeric or character (with default): optional position coordinates or keyword (e.g. "topright") for the legend to be plotted.
na.exclude	logical: indicating wether NA values are removed before plotting from the input data set
...	further arguments and graphical parameters passed to plot.

Details

Procedure to test the accuracy of a measurement protocol to reliably determine the dose of a specific sample. Here, the natural signal is erased and a known laboratory dose administered which is treated as unknown. Then the De measurement is carried out and the degree of congruence between administered and recovered dose is a measure of the protocol's accuracy for this sample.
In the plot the normalised De is shown on the y-axis, i.e. obtained De/Given Dose.

Value

A plot is returned.

Function version

0.1.2 (2014-04-13 14:30:15)

Note

Further data and plot arguments can be added by using the appropiate R commands.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany), Michael Dietze, GFZ Potsdam (Germany),
R Luminescence Package Team

References

Wintle, A.G., Murray, A.S., 2006. A review of quartz optically stimulated luminescence characteristics and their relevance in single-aliquot regeneration dating protocols. Radiation Measurements, 41, 369-391.

See Also

plot

Examples

```
## read example data set and misapply them for this plot type
data(ExampleData.DeValues, envir = environment())

## plot values
plot_DRTResults(values = ExampleData.DeValues[7:11,],
given.dose = 2800, mtext = "Example data")

## plot values with legend
plot_DRTResults(values = ExampleData.DeValues[7:11,],
                given.dose = 2800,
                legend = "Test data set")

## create and plot two subsets with randomised values
x.1 <- ExampleData.DeValues[7:11,]
x.2 <- ExampleData.DeValues[7:11,] * c(runif(5, 0.9, 1.1), 1)

plot_DRTResults(values = list(x.1, x.2),
                given.dose = 2800)

## some more user-defined plot parameters
plot_DRTResults(values = list(x.1, x.2),
                given.dose = 2800,
                pch = c(2, 5),
                col = c("orange", "blue"),
                xlim = c(0, 8),
                ylim = c(0.85, 1.15),
                xlab = "Sample aliquot")

## plot the data with user-defined statistical measures as legend
plot_DRTResults(values = list(x.1, x.2),
```

```
                given.dose = 2800,
                summary = c("n", "mean.weighted", "sd"))

## plot the data with user-defined statistical measures as sub-header
plot_DRTResults(values = list(x.1, x.2),
                given.dose = 2800,
                summary = c("n", "mean.weighted", "sd"),
                summary.pos = "sub")

## plot the data grouped by preheat temperatures
plot_DRTResults(values = ExampleData.DeValues[7:11,],
                given.dose = 2800,
                preheat = c(200, 200, 200, 240, 240))

## plot two data sets grouped by preheat temperatures
plot_DRTResults(values = list(x.1, x.2),
                given.dose = 2800,
                preheat = c(200, 200, 200, 240, 240))

## plot the data grouped by preheat temperatures as boxplots
plot_DRTResults(values = ExampleData.DeValues[7:11,],
                given.dose = 2800,
                preheat = c(200, 200, 200, 240, 240),
                boxplot = TRUE)
```

`plot_GrowthCurve` *Fit and plot a growth curve for luminescence data (Lx/Tx against dose)*

Description

A dose response curve is produced for luminescence measurements using a regenerative protocol.

Usage

```
plot_GrowthCurve(sample, na.exclude = TRUE, main = "Growth curve",
   fit.method = "EXP", fit.weights = TRUE, fit.includingRepeatedRegPoints = TRUE,
    fit.NumberRegPoints, fit.NumberRegPointsReal, fit.bounds = TRUE,
    NumberIterations.MC = 100, output.plot = TRUE, output.plotExtended = TRUE,
    cex.global = 1, ...)
```

Arguments

`sample`	`data.frame` (**required**): data frame with three columns for x=Dose,y=LxTx,z=LxTx.Error, y1=TnTx. The column for the test dose response is optional, but requires 'TnTx' as column name if used.
`na.exclude`	`logical` (with default): excludes `NA` values from the data set prior to any further operations.
`main`	`character` (with default): header of the plot.

`fit.method` character (with default): function used for fitting. Possible options are: `LIN`, `EXP`, `EXP OR LIN`, `EXP+LIN` or `EXP+EXP`. See details.

`fit.weights` logical (with default): option whether the fitting is done with or without weights. See details.

`fit.includingRepeatedRegPoints` logical (with default): includes repeated points for fitting (`TRUE/FALSE`).

`fit.NumberRegPoints` integer (optional): set number of regeneration points manually. By default the number of all (!) regeneration points is used automatically.

`fit.NumberRegPointsReal` integer (optional): if the number of regeneration points is provided manually, the value of the real, regeneration points = all points (repeated points) including reg 0, has to be inserted.

`fit.bounds` logical (with default): set lower fit bounds for all fitting parameters to 0. Limited for the use with the fit methods `EXP`, `EXP+LIN` and `EXP OR LIN`. Argument to be inserted for experimental application only!

`NumberIterations.MC` integer (with default): number of Monte Carlo simulations for error estimation. See details.

`output.plot` logical (with default): plot output (`TRUE/FALSE`).

`output.plotExtended` logical (with default): If `TRUE`, 3 plots on one plot area are provided: (1) growth curve, (2) histogram from Monte Carlo error simulation and (3) a test dose response plot. If `FALSE`, just the growth curve will be plotted. **Requires:** `output.plot = TRUE`.

`cex.global` numeric (with default): global scaling factor.

`...` Further arguments and graphical parameters to be passed. Note: Standard arguments will only be passed to the growth curve plot

Details

Fitting methods

For all options (except for the `LIN` and the `EXP OR LIN`), the nls function with the `port` algorithm is used.
`LIN`: fits a linear function to the data using lm:

$$y = m * x + n$$

`EXP`: try to fit a function of the form

$$y = a * (1 - exp(-(x + c)/b))$$

Parameters b and c are approximated by a linear fit using lm.
`EXP OR LIN`: works for some cases where an `EXP` fit fails. If the `EXP` fit fails, a `LIN` fit is done instead.
`EXP+LIN`: tries to fit an exponential plus linear function of the form:

$$y = a * (1 - exp(-(x + c)/b) + (g * x))$$

The De is calculated by iteration.
Note: In the context of luminescence dating, this function has no physical meaning. Therefore, no D0 value is returned.
`EXP+EXP`: tries to fit a double exponential function of the form

$$y = (a1 * (1 - exp(-(x)/b1))) + (a2 * (1 - exp(-(x)/b2)))$$

This fitting procedure is not robust against wrong start parameters and should be further improved.

Fit weighting
If the option `fit.weights = TRUE` is chosen, weights are calculated using provided signal errors (Lx/Tx error):

$$fit.weights = 1/error/(sum(1/error))$$

Error estimation using Monte Carlo simulation
Error estimation is done using a Monte Carlo (MC) simulation approach. A set of values is constructed by randomly drawing curve data from a normal distribution. The normal distribution is defined by the input values (mean = value, sd = value.error). Then, a growth curve fit is attempted for each dataset which results in new distribution of values. The sd of this distribution is the error of the De. With increasing iterations, the error value is becoming more stable. **Note:** It may take some calculation time with increasing MC runs, especially for the composed functions (`EXP+LIN` and `EXP+EXP`).
Each error estimation is done with the function of the chosen fitting method.

Subtitle information
To avoid plotting the subtitle information, provide an empty user mtext `mtext = ""`. To plot any other subtitle text, use `mtext`.

Value

`RLum.Results` object containing the De (De, De Error, D01 value, D02 value and Fit type) and fit object nls object for `EXP`, `EXP+LIN` and `EXP+EXP`. In case of a resulting linear fit when using `EXP OR LIN`, a lm object is returned. Additionally a plot is returned.

Function version

1.2.6 (2014-06-04 19:45:24)

Author(s)

Sebastian Kreutzer, Universite Bordeaux Montaigne (France), Michael Dietze, GFZ Potsdam (Germany),
R Luminescence Package Team

Examples

```
##(1) plot growth curve for a dummy data.set
data(ExampleData.LxTxData, envir = environment())
plot_GrowthCurve(LxTxData)
```

```
##(2) plot the growth curve only - uncomment to use

##pdf(file = "~/Desktop/Growth_Curve_Dummy.pdf", paper = "special")
plot_GrowthCurve(LxTxData)
##dev.off()

##(3) plot growth curve with pdf output - uncomment to use

##pdf(file = "~/Desktop/Growth_Curve_Dummy.pdf", paper = "special")
plot_GrowthCurve(LxTxData)
##dev.off()
```

`plot_Histogram` *Plot a histogram with a separate error plot*

Description

Function plots a predefined histogram with an accompanying error plot as suggested by Rex Galbraith at the UK LED in Oxford 2010.

Usage

```
plot_Histogram(values, na.exclude = TRUE, mtext, cex.global,
    breaks, se, rug, normal_curve, summary, summary.pos, colour,
    ...)
```

Arguments

values	data.frame or RLum.Results object (required): for data.frame: two columns: De (values[,1]) and De error (values[,2])
na.exclude	logical (with default): excludes NA values from the data set prior to any further operations.
mtext	character (optional): further sample information (mtext).
cex.global	numeric (with default): global scaling factor.
breaks	(with default): sets breakpoints for histogram. Works as in hist.
se	logical (optional): plots standard error points over the histogram, default is FALSE.
rug	logical (optional): adds rugs to the histogram, default is TRUE.
normal_curve	logical (with default): adds a normal curve to the histogram. Mean and sd are calculated from the input data. More see details section.
summary	character (optional): adds numerical output to the plot. Can be one or more out of: "n" (number of samples), "mean" (mean De value), "median" (median of the De values), "kdemax" (maximum value of probability density function), "sdrel" (relative standard deviation), "sdabs" (absolute standard deviation), "serel" (relative standard error) and "seabs" (absolute standard deviation).

summary.pos — numeric (with default): optional position coordinates for the statistical summary. Y-coordinate refers to the right hand y-axis.

colour — numeric or character (with default): optional vector of length 4 which specifies the colours of the following plot items in exactly this order: histogram bars, rug lines, normal distribution curve and standard error points (e.g., c("grey", "black", "red", "grey")).

... — further arguments and graphical parameters passed to plot. If y-axis labels are provided, these must be specified as a vector of length 2 since the plot features two axes (e.g. ylab = c("axis label 1", "axis label 2")). Y-axes limits (ylim) must be provided as vector of length four, with the first two elements specifying the left axes limits and the latter two elements giving the right axis limits.

Details

If the normal curve is added, the y-axis in the histogram will show the probability density.

Function version

0.4.1 (2014-04-13 14:30:27)

Note

The input data is not restricted to a special type.

Author(s)

Michael Dietze (GFZ Potsdam),
Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

See Also

hist, plot

Examples

```
## load data
data(ExampleData.DeValues, envir = environment())
ExampleData.DeValues <-
  Second2Gray(values = ExampleData.DeValues, dose_rate = c(0.0438,0.0019))

## plot histogram the easiest way
plot_Histogram(ExampleData.DeValues)

## plot histogram with some more modifications
plot_Histogram(ExampleData.DeValues,
               rug = TRUE,
               normal_curve = TRUE,
               cex.global = 0.9,
```

```
pch = 2,
colour = c("grey", "black", "blue", "green"),
summary = c("n", "mean", "sdrel"),
summary.pos = "topleft",
main = "Histogram of De-values",
mtext = "Example data set",
ylab = c(expression(paste(D[e], " distribution")),
         "Error"),
xlim = c(100, 250),
ylim = c(0, 0.08, 50, 200))
```

---

plot_KDE *Plot kernel density estimate with statistics*

Description

Plot a kernel density estimate of measurement values in combination with the actual values and associated error bars in ascending order. Optionally, statistical measures such as mean, median, standard deviation, standard error and quartile range can be provided visually and numerically.

Usage

```
plot_KDE(data, na.exclude = TRUE, weights = FALSE, values.cumulative = TRUE,
    centrality, dispersion, stats, stats.pos = "sub", polygon.col,
    order = TRUE, bw = "nrd0", output = FALSE, ...)
```

Arguments

data	data.frame or RLum.Results object (required): for data.frame: two columns: De (values[,1]) and De error (values[,2]). For plotting multiple data sets, these must be provided as list (e.g. list(dataset1, dataset2)).
na.exclude	logical (with default): exclude NA values from the data set prior to any further operations.
weights	logical (with default): calculate the KDE with De-errors as weights. Attention, using errors as weights will result in a plot similar to a a probability density plot, with all ambiguities related to this plot type!
values.cumulative	logical (with default): show cumulative individual data.
centrality	character: measure(s) of centrality, used for plotting vertical lines of the respective measure. Can be one out of "mean", "median", "mean.weighted", "median.weighted" and "kdemax".
dispersion	character: measure of dispersion, used for drawing the polygon that depicts the dose distribution. One out of "sd" (standard deviation),"2sd" (2 standard deviations) "qr" (quartile range).

stats	character (optional): add numerical output to the plot. Can be one or more out of: "n" (number of samples), "mean" (mean De value), "mean.weighted" (error-weighted mean), "median.weighted" (error-weighted median), "median" (median of the De values), "kdemax" (maximum value of probability density function), "kurtosis" (kurtosis), "skewness" (skewness), "sdrel" (relative standard deviation in percent), "sdabs" (absolute standard deviation), "serel" (relative standard error) and "seabs" (absolute standard error).
stats.pos	numeric or character (with default): optional position coordinates or keyword for the statistical summary. Y-coordinate refers to the left y-axis.
polygon.col	character or numeric (with default): colour of the polygon showing the dose dispersion around the central value. Only relevant if dispersion is specified.
order	logical: Order data in ascending order.
bw	character (with default): bin-width, chose a numeric value for manual setting.
output	logical: Optional output of numerical plot parameters. These can be useful to reproduce similar plots. Default is FALSE.
...	further arguments and graphical parameters passed to plot.

Details

The function allow passing several plot arguments, such as main, xlab, cex. However, as the figure is an overlay of two separate plots, ylim must be specified in the order: c(ymin_axis1, ymax_axis1, ymin_axis2, ymax_axis2) when using the cumulative values plot option. Similarly, if other than the default colours are desired, the argument col must be provided with colours in the following order: probability density function, De values, De error bars, sd or qr polygon. The line type (lty) for additional measures of centrality will cycle through the default values (1, 2, ...) by default, i.e. KDE line solid, further vertical lines dashed, dotted, dash-dotted and so on. To change this behaviour specify the desired order of line types (e.g. lty = c(1, 3, 2, 5)). See examples for some further explanations. For details on the calculation of the bin-width (parameter bw) see density.

Function version

3.2 (2014-05-30 11:58:30)

Note

The plot output is no 'PD' plot (cf. the discussion of Berger and Galbraith in Ancient TL; see references)!

Author(s)

Michael Dietze (GFZ Potsdam), Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

See Also

density, plot

Examples

```
## read example data set
data(ExampleData.DeValues, envir = environment())
ExampleData.DeValues <-
  Second2Gray(values = ExampleData.DeValues, dose_rate = c(0.0438,0.0019))

## create plot straightforward
plot_KDE(data = ExampleData.DeValues)

## create plot with logarithmic x-axis
plot_KDE(data = ExampleData.DeValues,
         log = "x")

## create plot with user-defined labels and axes limits
plot_KDE(data = ExampleData.DeValues,
         main = "Dose distribution",
         xlab = "Dose [s]",
         ylab = c("KDE estimate", "Cumulative dose value"),
         xlim = c(100, 250),
         ylim = c(0, 0.08, 0, 30))

## create plot with centrality lines and distribution polygons
plot_KDE(data = ExampleData.DeValues,
         ylim = c(0, 0.08, 0, 35),
         centrality = c("median", "mean"),
         dispersion = "sd",
         polygon.col = "lightblue")

## create plot with statistical summary below header
plot_KDE(data = ExampleData.DeValues,
         stats = c("n", "median", "skewness", "qr"))

## create plot with statistical summary as legend
plot_KDE(data = ExampleData.DeValues,
         stats = c("n", "mean", "sdrel", "seabs"),
         stats.pos = "topleft")

## split data set into sub-groups, one is manipulated, and merge again
data.1 <- ExampleData.DeValues[1:15,]
data.2 <- ExampleData.DeValues[16:25,] * 1.3
data.3 <- list(data.1, data.2)

## create plot with two subsets straightforward
plot_KDE(data = data.3)

## create plot with two subsets and summary legend at user coordinates
plot_KDE(data = data.3,
         stats = c("n", "median", "skewness"),
         stats.pos = c(110, 0.07),
         col = c("blue", "orange"))

## example of how to use the numerical output of the function
```

```
## return plot output to draw a thicker KDE line
KDE <- plot_KDE(data = ExampleData.DeValues,
                output = TRUE)

## read out coordinates of KDE graph
KDE.x <- KDE$De.density[[1]]$x
KDE.y <- KDE$De.density[[1]]$y

## transform y-values to right y-axis dimensions
KDE.y <- KDE.y / max(KDE.y) * (nrow(ExampleData.DeValues) - 1) + 1

## draw the KDE line
lines(x = KDE.x,
      y = KDE.y,
      lwd = 3)
```

`plot_RadialPlot` *Function to create a Radial Plot*

Description

A Galbraith's radial plot is produced on a logarithmic or a linear scale.

Usage

```
plot_RadialPlot(data, na.exclude = TRUE, negatives = "remove",
    log.z = TRUE, central.value, centrality = "mean.weighted",
    mtext, summary, summary.pos, legend, legend.pos, stats, plot.ratio,
    bar.col, y.ticks = TRUE, grid.col, line, line.col, line.label,
    output = FALSE, ...)
```

Arguments

`data`	`data.frame` or `RLum.Results` object (required): for `data.frame` two columns: De (`data[,1]`) and De error (`data[,2]`). To plot several data sets in one plot, the data sets must be provided as `list`, e.g. `list(data.1, data.2)`.
`na.exclude`	`logical` (with default): excludes `NA` values from the data set prior to any further operations.
`negatives`	`character` (with default): rule for negative values. Default is `"remove"` (i.e. negative values are removed from the data set).
`log.z`	`logical` (with default): Option to display the z-axis in logarithmic scale. Default is `TRUE`.
`central.value`	`numeric`: User-defined central value, primarily used for horizontal centering of the z-axis.
`centrality`	`character` (with default): measure of centrality, used for automatically centering the plot and drawing the central line. Can be one out of `"mean"`, `"median"`, `"mean.weighted"`, and `"median.weighted"`. Default is `"mean.weighted"`.

mtext	character: additional text below the plot title.
summary	character (optional): adds numerical output to the plot. Can be one or more out of: "n" (number of samples), "mean" (mean De value), "mean.weighted" (error-weighted mean), "median" (median of the De values), "sdrel" (relative standard deviation in percent), "sdabs" (absolute standard deviation), "serel" (relative standard error), "seabs" (absolute standard error) and "in.ci" (percent of samples in confidence interval, e.g. 2-sigma).
summary.pos	numeric or character (with default): optional position coordinates or keyword (e.g. "topright") for the statistical summary. Alternatively, the keyword "sub" may be specified to place the summary below the plot header. However, this latter option is only possible if mtext is not used.
legend	character vector (optional): legend content to be added to the plot.
legend.pos	numeric or character (with default): optional position coordinates or keyword (e.g. "topright") for the legend to be plotted.
stats	character: additional labels of statistically important values in the plot. One or more out of the following: "min", "max", "median".
plot.ratio	numeric: User-defined plot area ratio (i.e. curvature of the z-axis). If omitted, the default value (4.5/5.5) is used and modified automatically to optimise the z-axis curvature. The parameter should be decreased when data points are plotted outside the z-axis or when the z-axis gets too elliptic.
bar.col	character or numeric (with default): colour of the bar showing the 2-sigma range around the central value. To disable the bar, use "none". Default is "grey".
y.ticks	logical: Option to hide y-axis labels. Useful for data with small scatter.
grid.col	character or numeric (with default): colour of the grid lines (originating at [0,0] and stretching to the z-scale). To disable grid lines, use "none". Default is "grey".
line	numeric: numeric values of the additional lines to be added.
line.col	character or numeric: colour of the additional lines.
line.label	character: labels for the additional lines.
output	logical: Optional output of numerical plot parameters. These can be useful to reproduce similar plots. Default is FALSE.
...	Further plot arguments to pass. xlab must be a vector of length 2, specifying the upper and lower x-axes labels.

Details

Details and the theoretical background of the radial plot are given in the cited literature. This function is based on an S script of Rex Galbraith. To reduce the manual adjustments, the function has been rewritten. Thanks to Rex Galbraith for useful comments on this function.
Plotting can be disabled by adding the argument plot = "FALSE", e.g. to return only numeric plot output.

Earlier versions of the Radial Plot in this package had the 2-sigma-bar drawn onto the z-axis. However, this might have caused misunderstanding in that the 2-sigma range may also refer to the

z-scale, which it does not! Rather it applies only to the x-y-coordinate system (standardised error vs. precision). A spread in doses or ages must be drawn as lines originating at zero precision (x0) and zero standardised estimate (y0). Such a range may be drawn by adding lines to the radial plot (`line`, `line.col`, `line.label`, cf. examples).

Value

Returns a plot object.

Function version

0.5.2 (2014-06-05 13:42:41)

Author(s)

Michael Dietze, GFZ Potsdam (Germany), Sebastian Kreutzer, JLU Giessen (Germany)
Based on a rewritten S script of Rex Galbraith, 2010
R Luminescence Package Team

References

Galbraith, R.F., 1988. Graphical Display of Estimates Having Differing Standard Errors. Technometrics, 30 (3), 271-281.

Galbraith, R.F., 1990. The radial plot: Graphical assessment of spread in ages. International Journal of Radiation Applications and Instrumentation. Part D. Nuclear Tracks and Radiation Measurements, 17 (3), 207-214.

Galbraith, R. & Green, P., 1990. Estimating the component ages in a finite mixture. International Journal of Radiation Applications and Instrumentation. Part D. Nuclear Tracks and Radiation Measurements, 17 (3) 197-206.

Galbraith, R.F. & Laslett, G.M., 1993. Statistical models for mixed fission track ages. Nuclear Tracks And Radiation Measurements, 21 (4), 459-470.

Galbraith, R.F., 1994. Some Applications of Radial Plots. Journal of the American Statistical Association, 89 (428), 1232-1242.

Galbraith, R.F., 2010. On plotting OSL equivalent doses. Ancient TL, 28 (1), 1-10.

Galbraith, R.F. & Roberts, R.G., 2012. Statistical aspects of equivalent dose and error calculation and display in OSL dating: An overview and some recommendations. Quaternary Geochronology, 11, 1-27.

See Also

`plot`, `plot_KDE`, `plot_Histogram`

Examples

```
## load example data
data(ExampleData.DeValues, envir = environment())
ExampleData.DeValues <-
  Second2Gray(values = ExampleData.DeValues, dose_rate = c(0.0438,0.0019))
```

```
## plot the example data straightforward
plot_RadialPlot(data = ExampleData.DeValues)

## now with linear z-scale
plot_RadialPlot(data = ExampleData.DeValues,
                log.z = FALSE)

## now with output of the plot parameters
plot1 <- plot_RadialPlot(data = ExampleData.DeValues,
                         log.z = FALSE,
                         output = TRUE)
plot1
plot1$zlim

## now with adjusted z-scale limits
plot_RadialPlot(data = ExampleData.DeValues,
                log.z = FALSE,
                zlim = c(100, 200))

## now the two plots with serious but seasonally changing fun
#plot_RadialPlot(data = data.3, fun = TRUE)

## now with user-defined central value, in log-scale again
plot_RadialPlot(data = ExampleData.DeValues,
                central.value = 150)

## now with legend, colour, different points and smaller scale
plot_RadialPlot(data = ExampleData.DeValues,
                legend.text = "Sample 1",
                col = "tomato4",
                bar.col = "peachpuff",
                pch = "R",
                cex = 0.8)

## now without 2-sigma bar, y-axis, grid lines and central value line
plot_RadialPlot(data = ExampleData.DeValues,
                bar.col = "none",
                grid.col = "none",
                y.ticks = FALSE,
                lwd = 0)

## now with user-defined axes labels
plot_RadialPlot(data = ExampleData.DeValues,
                xlab = c("Data error [%]",
                         "Data precision"),
                ylab = "Scatter",
                zlab = "Equivalent dose [Gy]")

## now with minimum, maximum and median value indicated
plot_RadialPlot(data = ExampleData.DeValues,
                central.value = 150,
                stats = c("min", "max", "median"))
```

```
## now with a brief statistical summary
plot_RadialPlot(data = ExampleData.DeValues,
                summary = c("n", "in.ci"))

## now with another statistical summary as subheader
plot_RadialPlot(data = ExampleData.DeValues,
                summary = c("mean.weighted", "median"),
                summary.pos = "sub")

## now the data set is split into sub-groups, one is manipulated
data.1 <- ExampleData.DeValues[1:15,]
data.2 <- ExampleData.DeValues[16:25,] * 1.3

## now a common dataset is created from the two subgroups
data.3 <- list(data.1, data.2)

## now the two data sets are plotted in one plot
plot_RadialPlot(data = data.3)

## now with some graphical modification
plot_RadialPlot(data = data.3,
                col = c("darkblue", "darkgreen"),
                bar.col = c("lightblue", "lightgreen"),
                pch = c(2, 6),
                summary = c("n", "in.ci"),
                summary.pos = "sub",
                legend = c("Sample 1", "Sample 2"))
```

`plot_Risoe.BINfileData` *Plot single luminescence curves from a BIN file object*

Description

Plots single luminescence curves from an object returned by the readBIN2R function.

Usage

```
plot_Risoe.BINfileData(BINfileData, position, run, set, sorter = "POSITION",
    ltype = c("IRSL", "OSL", "TL", "RIR", "RBR", "RL"), curve.transformation,
    dose_rate, temp.lab, cex.global = 1, ...)
```

Arguments

`BINfileData`	Risoe.BINfileData-class (**required**): requires an S4 object returned by the readBIN2R function.
`position`	vector (optional): option to limit the plotted curves by position (e.g. `position = 1`, `position = c(1,3,5)`).

`run`	vector (optional): option to limit the plotted curves by run (e.g., `run = 1`, `run = c(1,3,5)`).
`set`	vector (optional): option to limit the plotted curves by set (e.g., `set = 1`, `set = c(1,3,5)`).
`sorter`	character (with default): the plot output can be ordered by "POSITION","SET" or "RUN". POSITION, SET and RUN are options defined in the Risoe Sequence Editor.
`ltype`	character (with default): option to limit the plotted curves by the type of luminescence stimulation. Allowed values: `"IRSL"`, `"OSL"`,`"TL"`, `"RIR"`, `"RBR"` (corresponds to LM-OSL), `"RL"`. All type of curves are plotted by default.
`curve.transformation`	character (optional): allows transforming CW-OSL and CW-IRSL curves to pseudo-LM curves via transformation functions. Allowed values are: `CW2pLM`, `CW2pLMi`, `CW2pHMi` and `CW2pPMi`. See details.
`dose_rate`	numeric (optional): dose rate of the irradition source at the measurement date. If set, the given irradiation dose will be shown in Gy. See details.
`temp.lab`	character (optional): option to allow for different temperature units. If no value is set deg. C is chosen.
`cex.global`	numeric (with default): global scaling factor.
`...`	further undocumented plot arguments.

Details

Nomenclature

The nomenclature used for the function (e.g., ltype, position) are taken from the Analyst manual (Duller, 2007, p. 42):

[,1]	ID	: Unique record ID (same ID as in slot `DATA`)	`numeric`
[,2]	SEL	: Record selection	`logical`
[,3]	VERSION	: Data format version number	`raw`
[,4]	LENGTH	: Length of this record	`integer`
[,5]	PREVIOUS	: Length of previous record	`integer`
[,6]	NPOINTS	: Number of data points in the record	`integer`
[,7]	LTYPE	: Luminescence type	`factor`
[,8]	LOW	: Low (temperature, time, wavelength)	`numeric`
[,10]	HIGH	: High (temperature, time, wavelength)	`numeric`
[,11]	RATE	: Rate (heating rate, scan rate)	`numeric`
[,12]	TEMPERATURE	: Sample temperature	`integer`
[,13]	XCOORD	: X position of a single grain	`integer`
[,14]	YCOORD	: Y position of a single grain	`integer`
[,15]	TOLDELAY	: TOL 'delay' channels	`integer`
[,16]	TOLON	: TOL 'on' channels	`integer`
[,17]	TOLOFF	: TOL 'off' channels	`integer`
[,18]	POSITION	: Carousel position	`integer`

[,19]	RUN	: Run number	integer
[,20]	TIME	: Data collection time (hh-mm-ss)	factor
[,21]	DATA	: Data collection date (dd-mm-yy)	factor
[,22]	SEQUENCE	: Sequence name	factor
[,23]	USER	: User name	factor
[,24]	DTYPE	: Data type	factor
[,25]	IRR_TIME	: Irradiation time	numeric
[,26]	IRR_TYPE	: Irradiation type (alpha, beta or gamma)	integer
[,27]	IRR_UNIT	: Irradiation unit (Gy, Rads, secs, mins, hrs)	integer
[,28]	BL_TIME	: Bleaching time	numeric
[,29]	BL_UNIT	: Bleaching unit (mJ, J, secs, mins, hrs)	integer
[,30]	AN_TEMP	: Annealing temperature	numeric
[,31]	AN_TIME	: Annealing time	numeric
[,32]	NORM1	: Normalisation factor (1)	numeric
[,33]	NORM2	: Normalisation factor (2)	numeric
[,34]	NORM3	: Normalisation factor (3)	numeric
[,35]	BG	: Background level	numeric
[,36]	SHIFT	: Number of channels to shift data	integer
[,37]	SAMPLE	: Sample name	factor
[,38]	COMMENT	: Comment	factor
[,39]	LIGHTSOURCE	: Light source	factor
[,40]	SET	: Set Number	integer
[,41]	TAG	: Tag	integer
[,42]	GRAIN	: Grain number	integer
[,43]	LPOWER	: Optical Stimulation Power	numeric
[,44]	SYSTEMID	: System ID	integer

curve.transformation

This argument allows transforming continuous wave (CW) curves to pseudo (linear) modulated curves. For the transformation, the functions of the package are used. Currently, it is not possible to pass further arguments to the transformation functions. The argument works only for `ltype` `OSL` and `IRSL`.

Irradiation time

Plotting the irradiation time (s) or the given dose (Gy) requires that the variable `IRR_TIME` has been set within the BIN-file. This is normally done by using the 'Run Info' option within the Sequence Editor or by editing in R.

Value

Returns a plot.

Function version

0.4.1 (2014-04-13 14:30:45)

Note

The function has been successfully tested for the Sequence Editor file output version 3 and 4.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany), Michael Dietze, GFZ Potsdam (Germany)
R Luminescence Package Team

References

Duller, G., 2007. Analyst. pp. 1-45.

See Also

readBIN2R, CW2pLM, CW2pLMi, CW2pPMi, CW2pHMi

Examples

```
##load data
data(ExampleData.BINfileData, envir = environment())

##plot all curves from the first position to the desktop
#pdf(file = "~/Desktop/CurveOutput.pdf", paper = "a4", height = 11, onefile = TRUE)

##example - load from *.bin file
#BINfile<-"[your path]"
#BINfileData<-readBIN2R(BINfile)

#par(mfrow = c(4,3), oma = c(0.5,1,0.5,1))
#plot_Risoe.BINfileData(CWOSL.SAR.Data,position = 1)
#mtext(side = 4, BINfile, outer = TRUE, col = "blue", cex = .7)
#dev.off()
```

---

plot_RLum *General plot function for RLum S4 class objects*

Description

Function calls object specific plot functions for RLum S4 class objects.

Usage

```
plot_RLum(object, ...)
```

Arguments

object	RLum (**required**): S4 object of class RLum
...	further arguments and graphical parameters that will be passed to the specific plot functions

Details

The function provides a generalised access point for plotting specific RLum objects.
Depending on the input object, the corresponding plot function will be selected. Allowed arguments can be found in the documentations of each plot function.

object		corresponding plot function
RLum.Data.Curve	:	plot_RLum.Data.Curve
RLum.Analysis	:	plot_RLum.Analysis

Value

Returns a plot.

Function version

0.1 (2014-04-13 14:31:12)

Note

The provided plot output depends on the input object.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

References

#

See Also

plot_RLum.Data.Curve, RLum.Data.Curve

Examples

```
#load Example data
data(ExampleData.CW_OSL_Curve, envir = environment())

#transform data.frame to RLum.Data.Curve object
temp <- as(ExampleData.CW_OSL_Curve, "RLum.Data.Curve")

#plot RLum object
```

```
plot_RLum(temp)
```

`plot_RLum.Analysis` *Plot function for an RLum.Analysis S4 class object*

Description

The function provides a standardised plot output for curve data of an RLum.Analysis S4 class object

Usage

```
plot_RLum.Analysis(object, nrows = 3, ncols = 2, ...)
```

Arguments

`object`	`RLum.Analysis` (**required**): S4 object of class `RLum.Analysis`
`nrows`	`integer` (with default): sets number of rows for plot output
`ncols`	`integer` (with default): sets number of columns for plot output
`...`	further arguments and graphical parameters will be passed to the `plot` function.

Details

The function produces a multiple plot output. A file output is recommended (e.g., `pdf`).

Value

Returns multiple plots.

Function version

0.1.2 (2014-04-13 14:30:54)

Note

Not all arguments available for `plot` will be passed! Only plotting of `RLum.Data.Curve` and `RLum.Data.Spectrum` objects are currently supported.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

References

#

See Also

plot, plot_RLum, plot_RLum.Data.Curve

Examples

```
###load data
data(ExampleData.BINfileData, envir = environment())

##convert values for position 1
temp <- Risoe.BINfileData2RLum.Analysis(CWOSL.SAR.Data, pos=1)

##plot
plot_RLum.Analysis(temp)
```

---

plot_RLum.Data.Curve *Plot function for an RLum.Data.Curve S4 class object*

Description

The function provides a standardised plot output for curve data of an RLum.Data.Curve S4 class object

Usage

```
plot_RLum.Data.Curve(object, par.local = TRUE, ...)
```

Arguments

object	RLum.Data.Curve (**required**): S4 object of class RLum.Data.Curve
par.local	logical (with default): use local graphical parameters for plotting, e.g. the plot is shown in one column and one row. If par.local = FALSE, global parameters are inherited.
...	further arguments and graphical parameters that will be passed to the plot function

Details

Only single curve data can be plotted with this function. Arguments according to plot.

Value

Returns a plot.

Function version

0.1.3 (2014-04-13 14:30:59)

Note

Not all arguments of plot will be passed!

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

References

#

See Also

plot, plot_RLum

Examples

```
##plot curve data

#load Example data
data(ExampleData.CW_OSL_Curve, envir = environment())

#transform data.frame to RLum.Data.Curve object
temp <- as(ExampleData.CW_OSL_Curve, "RLum.Data.Curve")

#plot RLum.Data.Curve object
plot_RLum.Data.Curve(temp)
```

---

plot_RLum.Data.Spectrum

Plot function for an RLum.Data.Spectrum S4 class object

Description

The function provides a standardised plot output for spectrum data of an RLum.Data.Spectrum S4 class object

Usage

```
plot_RLum.Data.Spectrum(object, par.local = TRUE, plot.type = "contour",
    optical.wavelength.colours = TRUE, bg.channels, bin.rows = 1,
    bin.cols = 1, ...)
```

Arguments

object	RLum.Data.Spectrum (**required**): S4 object of class RLum.Data.Spectrum
par.local	logical (with default): use local graphical parameters for plotting, e.g. the plot is shown in one column and one row. If par.local = FALSE global parameters are inherited.
plot.type	character (with default): plot type, for 3D-plot use persp, or persp3d, for a 2D-plot contour, single or multiple.lines along the time or temperature axis Note: The use of persp3d will produce a dynamic 3D surface plot on the screen.
optical.wavelength.colours	logical (with default): use optical wavelength colour palette. Note: For this, the spectrum range is limited: c(350,750). Own colours can be set with the argument col.
bg.channels	vector (optional): defines channel for background subtraction If a vector is provided the mean of the channels is used for subtraction. Note: Background subtraction is applied prior to channel binning
bin.rows	integer (with defaul): allow summing-up wavelength channels (horizontal binning), e.g. bin.rows = 2 two channels are summed up
bin.cols	integer (with default): allow summing-up channel counts (vertical binning) for plotting, e.g. bin.cols = 2 two channels are summed up
...	further arguments and graphical parameters that will be passed to the plot function.

Details

Spectrum is visualised as 3D or 2D plot. Both plot types are based on internal R plot functions.

Arguments that will be passed to persp:

- shade: default is 0.4
- phi: default is 30
- theta: default is 30
- expand: default is 1
- ticktype: default is detailed

Further arguments that will be passed
xlab, ylab, zlab, xlim, ylim, zlim, main, mtext, pch, type, border, box lwd

plot.type = "single"
Per frame a single curve is returned. Frames are time or temperature steps.

plot.type = "multiple.lines"
All frames drawn in one frame.

Nomenclature

`xlim`: Limits values along the wavelength axis
`ylim`: Limits values along the time/temperature axis
`zlim`: Limits values along the count value axis

Value

Returns a plot.

Function version

0.2.5 (2014-05-21 16:58:16)

Note

Not all additional arguments (. . .) will be passed similarly!

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

References

#

See Also

`RLum.Data.Spectrum`, `plot`, `plot_RLum`, `persp`, `persp3d`, `contour`

Examples

```
##load example data
data(ExampleData.XSYG, envir = environment())

##(1)plot simple spectrum (2D) - contour
plot_RLum.Data.Spectrum(TL.Spectrum,
                        plot.type="contour",
                        xlim = c(310,750),
                        ylim = c(0,300),
                        bin.rows=10,
                        bin.cols = 1)

##(2) plot simple spectrum (2D) - multiple.lines (with ylim)
plot_RLum.Data.Spectrum(TL.Spectrum,
                        plot.type="multiple.lines",
                        xlim = c(310,750),
                        ylim = c(0,100),
                        bin.rows=10,
```

```
                        bin.cols = 1)

##(3) plot 3d spectrum (uncomment for usage)
# plot_RLum.Data.Spectrum(TL.Spectrum, plot.type="persp",
# xlim = c(310,750), ylim = c(0,300), bin.rows=10,
# bin.cols = 1)
```

---

readBIN2R *Import Risoe BIN-file into R*

Description

Import a *.bin or a *.binx file produced by a Risoe DA15 and DA20 TL/OSL reader into R.

Usage

```
readBIN2R(file, show.raw.values = FALSE, n.records, show.record.number = FALSE,
    txtProgressBar = TRUE, forced.VersionNumber)
```

Arguments

file	character (**required**): bin-file name (including path), e.g. [WIN]: `readBIN2R("C:/Desktop/test.bin")`, [MAC/LINUX]: `readBIN2R("/User/test/Desktop/test.bin")`
show.raw.values	logical (with default): shows raw values from BIN file for `LTYPE`, `DTYPE` and `LIGHTSOURCE` without translation in characters.
n.records	raw (optional): limits the number of imported records. Can be used in combination with `show.record.number` for debugging purposes, e.g. corrupt BIN files.
show.record.number	logical (with default): shows record number of the imported record, for debugging usage only.
txtProgressBar	logical (with default): enables or disables `txtProgressBar`.
forced.VersionNumber	integer (optional): allows to cheat the version number check in the function by own values for cases where the BIN-file version is not supported. Note: The usage is at own risk, only supported BIN-file versions have been tested.

Details

The binary data file is parsed byte by byte following the data structure published in the Appendices of the Analyst manual p. 42.

For the general BIN-file structure, the reader is referred to the Risoe website: `http://www.nutech.dtu.dk/`

Value

Returns an S4 Risoe.BINfileData-class object containing two slots:

METADATA	A data.frame containing all variables stored in the bin-file.
DATA	A list containing a numeric vector of the measured data. The ID corresponds to the record ID in METADATA.

Function version

0.7 (2014-04-13 14:31:53)

Note

The function has been successfully tested for BIN format versions 03, 04 and 06. The version number depends on the used Sequence Editor.

Other BIN format versions are currently not supported.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany), Margret C. Fuchs, AWI Postdam (Germany),
R Luminescence Package Team

References

Duller, G., 2007. Analyst. http://www.nutech.dtu.dk/english/~/media/Andre_Universitetsenheder/Nutech/Produkter%20og%20services/Dosimetri/radiation_measurement_instruments/tl_osl_reader/Manuals/analyst_manual_v3_22b.ashx

See Also

writeR2BIN, Risoe.BINfileData, readBin, merge_Risoe.BINfileData, txtProgressBar

Examples

```
##(1) import Risoe BIN-file to R (uncomment for usage)

#FILE <- file.choose()
#temp <- readBIN2R(FILE)
#temp
```

readXSYG2R *Import XSYG files to R*

Description

Imports XSYG files produced by a Freiberg Instrument lexsyg reader into R.

Usage

```
readXSYG2R(file, recalculate.TL.curves = TRUE, import = TRUE,
    txtProgressBar = TRUE)
```

Arguments

file	character (**required**): path and file name of the XSYG file.
recalculate.TL.curves	logical (with default): if set to TRUE, TL curves are returned as temperature against count values (see details for more information) Note: The option overwrites the time vs. count TL curve. Select FALSE to import the raw data delivered by the lexsyg.
import	logical (with default): if set to FALSE, only the XSYG file structure is shown.
txtProgressBar	logical (with default): enables TRUE or disables FALSE the progression bar during import

Details

How does the import function work?

The function uses the xml package to parse the file structure. Each sequence is subsequently translated into an RLum.Analysis object.

General structure XSYG format

```
<?xml?
<Sample>
<Sequence>
<Record>
<Curve name="first curve" />
<Curve name="curve with data">
x0 , y0 ; x1 , y1 ; x2 , y2 ; x3 , y3
</Curve>
</Record>
</Sequence>
</Sample>
```

So far, each XSYG file can only contain one <Sample></Sample>, but multiple sequences.

Each record may comprise several curves.

TL curve recalculation

On the FI lexsyg device TL curves are recorded as time against count values. Temperature values are monitored on the heating plate and stored in a separate curve (time vs. temperature). If the option `recalculate.TL.curves = TRUE` is chosen, the time values for each TL curve are replaced by temperature values.

Practically, this means combining two matrices (Time vs. Counts and Time vs. Temperature) with different row numbers by their time values. Three cases are considered:

HE: Heating element
PMT: Photomultiplier tube
Interpolation is done using the function approx

CASE (1): `nrow(matrix(PMT)) > nrow(matrix(HE))`

Missing temperature values from the heating element are calculated using time values from the PMT measurement.

CASE (2): `nrow(matrix(PMT)) < nrow(matrix(HE))`

Missing count values from the PMT are calculated using time values from the heating element measurement.

CASE (3): `nrow(matrix(PMT)) == nrow(matrix(HE))`

A new matrix is produced using temperature values from the heating element and count values from the PMT.

Note: Temperature values for spectrum curves are currently not supported. Please further note that due to the recalculation of the temperature values based on values delivered by the heating element, it may happen that mutiple count values exists for each temperature value and temperature values may also decrease during heating, not only increase.

Value

Using the option `import = FALSE`

A list consisting of two elements is shown:

`Sample` data.frame with information on file.

`Sequences` data.frame with information on the sequences stored in the XSYG file

.

Using the option `import = TRUE` **(default)**

A list is provided, the list elements contain:

`Sequence.Header`
data.frame with information on the sequence.

`Sequence.Object`
`RLum.Analysis` containing the curves.

Function version

0.3.1 (2014-06-06 14:35:51)

Note

This function is a beta version as the XSYG file format is not yet fully specified. Thus, further file operations (merge, export, write) should be done using the functions provided with the package `xml`.

So far, no image data import is provided!
Corresponding values in the XSXG file are skipped.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

References

Grehl, S., Kreutzer, S., Hoehne, M., 2013. Documentation of the XSYG file format. Unpublished Technical Note. Freiberg, Germany

Further reading

XML: http://en.wikipedia.org/wiki/XML

See Also

`xml`, `RLum.Analysis`, `RLum.Data.Curve`, `approx`

Examples

```
##(1) import XSYG file to R (uncomment for usage)

#FILE <- file.choose()
#temp <- readXSYG2R(FILE)
```

```
##(2) additional examples for pure XML import using the package XML
##    (uncomment for usage)

  ##import entire XML file
  #FILE <- file.choose()
  #temp <- xmlRoot(xmlTreeParse(FILE))

  ##search for specific subnodes with curves containing 'OSL'
  #getNodeSet(temp, "//Sample/Sequence/Record[@recordType = 'OSL']/Curve")
```

Risoe.BINfileData-class

Class "Risoe.BINfileData"

Description

S4 class object for luminescence data in R. The object is produced as output of the function readBIN2R.

Objects from the Class

Objects can be created by calls of the form new("Risoe.BINfileData", ...).

Slots

METADATA: Object of class "data.frame" containing the meta information for each curve.

DATA: Object of class "list" containing numeric vector with count data

.S3Class: Object of class "character"

Methods

show signature(object = "Risoe.BINfileData"): ...

set_Risoe.BINfileData signature(METADATA = "data.frame", DATA = "list"):
The Risoe.BINfileData is normally produced as output of the function readBIN2R. This construction method is intended for internal usage only.

get_Risoe.BINfileData signature(object = "Risoe.BINfileData"):
Formal get-method for Risoe.BINfileData object. It does not allow accessing the object directly, it is just showing a terminal message.

Version

0.3 (2013-06-02)

Note

Internal METADATA - object structure

#	Name	Data Type	V	Description
[,1]	ID	numeric	RLum	Unique record ID (same ID as in slot DATA)
[,2]	SEL	logic	RLum	Record selection
[,3]	VERSION	raw	03-06	BIN-file version number
[,4]	LENGTH	integer	03-06	Length of this record
[,5]	PREVIOUS	integer	03-06	Length of previous record
[,6]	NPOINTS	integer	03-06	Number of data points in the record
[,7]	RUN	integer	03-06	Run number
[,8]	SET	integer	03-06	Set number
[,9]	POSITION	integer	03-06	Position number
[,10]	GRAIN	integer	03-04	Grain number
[,11]	GRAINNUMBER	integer	06	Grain number
[,12]	CURVENO	integer	06	Curve number
[,13]	XCOORD	integer	03-06	X position of a single grain
[,14]	YCOORD	integer	03-06	Y position of a single grain
[,15]	SAMPLE	factor	03-06	Sample name
[,16]	COMMENT	factor	03-06	Comment name
[,17]	SYSTEMID	integer	03-06	Risoe system id
[,18]	FNAME	factor	06	File name (*.bin/*.binx)
[,19]	USER	facotr	03-06	User name
[,20]	TIME	character	03-06	Data collection time (hh-mm-ss)
[,21]	DATE	factor	03-06	Data collection date (ddmmyy)
[,22]	DTYPE	character	03-06	Data type
[,23]	BL_TIME	numeric	03-06	Bleaching time
[,24]	BL_UNIT	integer	03-06	Bleaching unit (mJ, J, secs, mins, hrs)
[,25]	NORM1	numeric	03-06	Normalisation factor (1)
[,26]	NORM2	numeric	03-06	Normalisation factor (2)
[,27]	NORM3	numeric	03-06	Normalisation factor (3)
[,28]	BG	numeric	03-06	Background level
[,29]	SHIFT	integer	03-06	Number of channels to shift data
[,30]	TAG	integer	03-06	Tag
[,31]	LTYPE	character	03-06	Luminescence type
[,32]	LIGHTSOURCE	character	03-06	Light source
[,33]	LPOWER	numeric	03-06	Optical stimulation power
[,34]	LIGHTPOWER	numeric	06	Optical stimulation power
[,35]	LOW	numeric	03-06	Low (temperature, time, wavelength)
[,36]	HIGH	numeric	03-06	High (temperature, time, wavelength)
[,37]	RATE	numeric	03-06	Rate (heating rate, scan rate)
[,38]	TEMPERATURE	integer	03-06	Sample temperature
[,39]	MEASTEMP	integer	06	Measured temperature
[,40]	AN_TEMP	numeric	03-06	Annealing temperature
[,41]	AN_TIME	numeric	03-06	Annealing time
[,42]	TOLDELAY	integer	03-06	TOL 'delay' channels
[,43]	TOLON	integer	03-06	TOL 'on' channels
[,44]	TOLOFF	integer	03-06	TOL 'off' channels
[,45]	IRR_TIME	numeric	03-06	Irradiation time
[,46]	IRR_TYPE	integer	03-06	Irradiation type (alpha, beta or gamma)
[,47]	IRR_UNIT	integer	03-04	Irradiation unit (Gy, Rads, secs, mins, hrs)

[,48]	IRR_DOSERATE	numeric	06	Irradiation dose rate (Gy/s)
[,49]	IRR_DOSERATEERR	numeric	06	Irradiation dose rate error (Gy/s)
[,50]	TIMESINCEIRR	integer	06	Time since irradiation (s)
[,51]	TIMETICK	numeric	06	Time tick for pulsing (s)
[,52]	ONTIME	integer	06	On-time for pulsing (in time ticks)
[,53]	STIMPERIOD	integer	06	Stimulation period (on+off in time ticks)
[,54]	GATE_ENABLED	raw	06	PMT signal gating enabled
[,55]	GATE_START	integer	06	Start gating (in time ticks)
[,56]	GATE_STOP	ingeter	06	Stop gating (in time ticks)
[,57]	PTENABLED	raw	06	Photon time enabled
[,58]	DTENABLED	raw	06	PMT dead time correction enabled
[,59]	DEADTIME	numeric	06	PMT dead time (s)
[,60]	MAXLPOWER	numeric	06	Stimulation power to 100 percent (mW/cm^2)
[,61]	XRF_ACQTIME	numeric	06	XRF acquisition time (s)
[,62]	XRF_HV	numeric	06	XRF X-ray high voltage (V)
[,63]	XRF_CURR	integer	06	XRF X-ray current (uA)
[,64]	XRF_DEADTIMEF	numeric	06	XRF dead time fraction
[,65]	SEQUENCE	character	03-04	Sequence name

V = BIN-file version (RLum means that it does not depend on a specific BIN version)

Note that the `Risoe.BINfileData` object combines all values from different versions from the BIN-file. Invalid values for a specific version are set to `NA`. Furthermore, the internal R data types do not necessarily match the required data types for the BIN-file data import! Data types are converted during data import.

LTYPES

[,0]	TL	: Thermoluminescence
[,1]	OSL	: Optically stimulated luminescence
[,2]	IRSL	: Infrared stimulated luminescence
[,3]	M-IR	: Infrared monochromator scan
[,4]	M-VIS	: Visible monochromator scan
[,5]	TOL	: Thermo-optical luminescence
[,6]	TRPOSL	: Time Resolved Pulsed OSL
[,7]	RIR	: Ramped IRSL
[,8]	RBR	: Ramped (Blue) LEDs
[,9]	USER	: User defined
[,10]	POSL	: Pulsed OSL
[,11]	SGOSL	: Single Grain OSL
[,12]	RL	: Radio Luminescence
[,13]	XRF	: X-ray Fluorescence

(information on the `LTYPE` kindly provided by Risoe, DTU Nutech)

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany)

References

Risoe DTU, 2013. The Sequence Editor User Manual - Feb 2013

http://www.nutech.dtu.dk/

See Also

plot_Risoe.BINfileData, readBIN2R, Risoe.BINfileData2RLum.Analysis

Examples

```
showClass("Risoe.BINfileData")
```

Risoe.BINfileData2RLum.Analysis

Convert Risoe.BINfileData object to an RLum.Analysis object

Description

Converts values from one specific position of a Risoe.BINfileData S4-class object to an RLum.Analysis object.

Usage

```
Risoe.BINfileData2RLum.Analysis(object, pos, run, set, ltype,
    protocol = "unknown")
```

Arguments

object	Risoe.BINfileData (**required**): Risoe.BINfileData object
pos	integer (**required**): position number of the Risoe.BINfileData object for which the curves are stored in the RLum.Analysis object.
run	vector, numeric (optional): run number from the measurement to limit the converted data set (e.g., run = c(1:48)).
set	vector, numeric (optional): set number from the measurement to limit the converted data set (e.g., set = c(1:48)).
ltype	vector, character (optional): curve type to limit the converted data. Allowed values are: IRSL, OSL, TL, RIR, RBR and USER
protocol	character (optional): sets protocol type for analysis object. Value may be used by subsequent analysis functions.

Details

The RLum.Analysis object requires a set of curves for specific further protocol analyses. However, the Risoe.BINfileData usually contains a set of curves for different aliquots and different protocol types that may be mixed up. Therefore, a conversion is needed.

Value

Returns an RLum.Analysis object.

Function version

0.1.1 (2014-04-13 14:32:02)

Note

The protocol argument of the RLum.Analysis object is set to 'unknown' if not stated otherwise.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany),
R Luminescence Package Team

References

#

See Also

Risoe.BINfileData, RLum.Analysis, readBIN2R

Examples

```
##load data
data(ExampleData.BINfileData, envir = environment())

##convert values for position 1
Risoe.BINfileData2RLum.Analysis(CWOSL.SAR.Data, pos = 1)
```

`Risoe.BINfileData2RLum.Data.Curve` *Convert an element from a Risoe.BINfileData object to an RLum.Data.Curve object*

Description

The function converts one specified single record from a Risoe.BINfileData object to an RLum.Data.Curve object.

Usage

```
Risoe.BINfileData2RLum.Data.Curve(object, id, pos, run, set)
```

Arguments

`object`	`Risoe.BINfileData` (**required**): `Risoe.BINfileData` object
`id`	`integer` (**required**): record id in the `Risoe.BINfileData` object of the curve that is to be stored in the `RLum.Data.Curve` object. If no value for id is provided, the record has to be specified by `pos`, `set` and `run`.
`pos`	`integer` (optional): record position number in the `Risoe.BINfileData` object of the curve that is to be stored in the `RLum.Data.Curve` object. If a value for `id` is provided, this argument is ignored.
`run`	`integer` (optional): record run number in the `Risoe.BINfileData` object of the curve that is to be stored in the `RLum.Data.Curve` object. If a value for `id` is provided, this argument is ignored.
`set`	`integer` (optional): record set number in the `Risoe.BINfileData` object of the curve that is to be stored in the `RLum.Data.Curve` object. If a value for `id` is provided, this argument is ignored.

Details

The function extracts all `METADATA` from the `Risoe.BINfileData` object and stores them in the `RLum.Data.Curve` object.

Value

Returns an `RLum.Data.Curve` object.

Function version

0.1 (2014-04-13 14:32:07)

Note

The function is intended for experimental usage. Normally, the function `Risoe.BINfileData2RLum.Analysis` should be used for the conversion.

Author(s)

Sebastian Kreutzer, Freiberg Instruments/JLU Giessen (Germany),
R Luminescence Package Team

References

#

See Also

Risoe.BINfileData2RLum.Analysis, set_RLum.Data.Curve, RLum.Data.Curve, RLum.Analysis, Risoe.BINfileData, plot_RLum

Examples

```
##get package example data
data(ExampleData.BINfileData, envir = environment())

##convert one record
Risoe.BINfileData2RLum.Data.Curve(CWOSL.SAR.Data, id = 1)
```

RLum-class *Class* "RLum"

Description

Abstract class for data in the package Luminescence

Objects from the Class

A virtual Class: No objects can be created from it.

Slots

`.S3Class`: Object of class `"character"`

Methods

No methods defined with class "RLum" in the signature.

Version

0.1 (2013-01-18)

Note

`RLum` is a virtual class.

Author(s)

Sebastian Kreutzer, 2013 (Freiberg Instruments/JLU Giessen, Germany)

References

#

See Also

RLum.Data, RLum.Analysis

Examples

```
showClass("RLum")
```

RLum.Analysis-class *Class* "RLum.Analysis"

Description

Object class containing analysis data for protocol analysis.

Objects from the Class

Objects can be created by calls of the form `new("RLum.Analysis", ...)`.

Slots

`records`**:** Object of class `"list"` containing objects of class RLum.Data

`protocol`**:** Object of class `"character"` describing the applied measurement protocol

`.S3Class`**:** Object of class `"character"`

Methods

show `signature(object = "RLum.Analysis")`: ...

set_RLum.Analysis `signature(records = "list", protocol = "character")`:

Construction method for `RLum.Analysis` object. The slot `protocol` is optional and predefined as `UNKNOWN` by default.

get_RLum.Analysis `signature(object = "RLum.Analysis",` reco

Accessor method for `RLum.Analysis` object.

The slots `record.id`, `recordType`, `curveType` and `RLum.type` are optional to allow for records limited by their id (list index number), their record type (e.g. `recordType = "OSL"`) or object type.

Example: curve type (e.g. `curveType = "predefined"` or `curveType ="measured"`)

The selection of a specific `RLum.type` **object superimposes the default selection.** Currently supported objects are: `RLum.Data.Curve` and `RLum.Data.Spectrum`

The argument `get.index = TRUE` just returns a `numeric vector` with the index of each element in the `RLum.Analysis` object.

The argument `keep.object` allowing returns an `RLum.Analysis` object instead of the single elements. Default is `keep.object = FALSE`.

get_structure.RLum.Analysis `signature(object = "RLum.Analysis")`: get meta structure of object as `data.frame`

length_RLum.Analysis `signature(object = "RLum.Analysis")`: returns length of the object, i.e., number of records in the object

Version

0.1.4 (2014-02-26)

Note

The method `get_structure.RLum.Analysis` is currently just avaiblable for objects containing `RLum.Data.Curve`.

Author(s)

Sebastian Kreutzer, Freiberg Instruments/JLU Giessen (Germany)

References

#

See Also

`Risoe.BINfileData2RLum.Analysis`, `Risoe.BINfileData`, `RLum`

Examples

```
showClass("RLum.Analysis")

## usage of get_RLum.Analysis() with returining an RLum.Analysis object
#  get_RLum.Analysis(object, keep.object = TRUE)
```

RLum.Data-class *Class* "RLum.Data"

Description

Generalized virtual data class for luminescence data.

Objects from the Class

A virtual Class: No objects can be created from it.

Slots

`.S3Class`**:** Object of class `"character"`

Methods

No methods defined with class "RLum.Data" in the signature.

Version

0.1 (2013-01-18)

Note

Just a virtual class.

Author(s)

Sebastian Kreutzer, 2013 (Freiberg Instruments/JLU Giessen, Germany)

References

#

See Also

`RLum`, `RLum.Data.Curve`, `RLum.Data.Spectrum`

Examples

```
showClass("RLum.Data")
```

RLum.Data.Curve-class *Class* "RLum.Data.Curve"

Description

Class for luminescence curve data.

Objects from the Class

Objects can be created by calls of the form `new("RLum.Data.Curve", ...)`.

Slots

`recordType`: Object of class `"character"` containing the type of the curve (e.g. "TL" or "OSL")

`curveType`: Object of class `"character"` containing curve type, allowed values are `measured` or `predefined`

`data`: Object of class `"matrix"` containing curve x and y data

`info`: Object of class `"list"` containing further meta information objects

`.S3Class`: Object of class `"character"`

Extends

Class "RLum.Data".

Methods

coerce `signature(from = "data.frame", to = "RLum.Data.Curve")`
`signature(from = "matrix", to = "RLum.Data.Curve")`
Furthermore, lossy coercing is possible from `RLum.Data.Curve` to:

`data.frame`, `matrix`

show `signature(object = "RLum.Data.Curve")`: ...

set_RLum.Data.Curve `signature(recordType = "character", curveType = "character", data = "matrix", inf`
Construction method for `RLum.Data.Curve` object. The slot `info` is optional and predefined as empty `list` by default.

get_RLum.Data.Curve `signature(object = "RLum.Data.Curve", info.object = "character")`:
Accessor method for `RLum.Data.Curve` object. The argument `info.object` is optional to directly access the info elements. If no info element name is provided, the raw curve data (`matrix`) will be returned.

Version

0.1.2 (2013-11-22)

Note

The class should only contain data for a single curve. For additional elements the slot `info` can be used (e.g. providing additional heating ramp curve).

Author(s)

Sebastian Kreutzer Freiberg Instruments/JLU Giessen (Germany)

References

#

See Also

RLum, RLum.Data, plot_RLum

Examples

```
showClass("RLum.Data.Curve")
```

RLum.Data.Spectrum-class

Class "RLum.Data.Spectrum"

Description

Class for luminescence spectra data (TL/OSL/RF).

Objects from the Class

Objects can be created by calls of the form `new("RLum.Data.Spectrum", ...)`.

Slots

`recordType`**:** Object of class `"character"` containing the type of the curve (e.g. "TL" or "OSL")

`curveType`**:** Object of class `"character"` containing curve type, allowed values are `measured` or `predefined`

`data`**:** Object of class `"matrix"` containing spectrum (count) values.

row labels indicating wavelength/pixel values
column labels temperature or time values.

`info`**:** Object of class `"list"` containing further meta information objects

`.S3Class`**:** Object of class `"character"`

Extends

Class "RLum.Data", directly.

Methods

coerce `signature(from = "data.frame", to = "RLum.Data.Spectrum")`
`signature(from = "matrix", to = "RLum.Data.Spectrum")`
Furthermore, lossy coercing is possible from `RLum.Data.Spectrum` to:

`data.frame`, `matrix`

show `signature(object = "RLum.Data.Spectrum")`: ...

set_RLum.Data.Spectrum `signature(recordType = "character", curveType = "character", data = "matrix",`
Construction method for `RLum.Data.Spectrum` object. The slot `info` is optional and predefined as empty `list` by default.

get_RLum.Data.Spectrum `signature(object = "RLum.Data.Spectrum", info.object = "character")`:
Accessor method for `RLum.Data.Spectrum` object. The argument `info.object` is optional to directly access the info elements. If no info element name is provided, the raw curve data (`matrix`) will be returned.

Version

0.1 (2013-11-23)

Note

The class should only contain data for a single spectra data set. For additional elements the slot `info` can be used.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany)

References

#

See Also

`RLum`, `RLum.Data`, `plot_RLum`

Examples

```
showClass("RLum.Data.Spectrum")

##show example data (uncomment for usage)
# data(ExampleData.XSYG, envir = environment())
# TL.Spectrum
```

RLum.Results-class *Class* "RLum.Results"

Description

Object class contains results data from functions.

Objects from the Class

Objects can be created by calls of the form `new("RLum.Results", ...)`.

Slots

`originator`: Object of class `"character"` containing name of the producing function

`data`: Object of class `"list"` containing output data

`.S3Class`: Object of class `"character"`

Methods

validObject `signature(object = "RLum.Results")`: validates object depending on the `originator` argument

show `signature(object = "RLum.Results")`: ...

set_RLum.Results `signature(originator = "character", data = "list")`:
Construction method for `RLum.Results` object. The slot `originator` is optional and predefined as the function that calls the function `set_RLum.Results`.

get_RLum.Results `signature(object = "RLum.Results", data.object = "character")`:
accessor method for `RLum.Results` object. The argument `data.object` allows directly accessing objects delivered within the slot `data`. If no `data.object` is specified, a preselected object is returned. The default return object depends on the object originator (e.g. `fit_LMCurve`).

merge_RLum.Results `signature(object.list = "list")`: merge method for `RLum.Results` objects. The argument `object.list` requires a list of `RLum.Results` objects.
Merging is done by appending similar elements to the first object of the input list.

Version

0.2.2 (2013-11-21)

Note

The class is intended to store results from functions to be used by other functions. The data in the object should always be accessed by the method `get_RLum.Results`.

Author(s)

Sebastian Kreutzer, 2013 (Freiberg Instruments/JLU Giessen, Germany)

References

#

See Also

RLum

Examples

```
showClass("RLum.Results")
```

Second2Gray *Converting values from seconds (s) to gray (Gy)*

Description

Conversion of absorbed radiation dose in seconds (s) to the SI unit gray (Gy) including error propagation. Normally used for equivalent dose data.

Usage

```
Second2Gray(values, dose_rate, method = "gaussian")
```

Arguments

values	data.frame (**required**): measured data (values[,1]) and data error (values [,2])
dose_rate	vector (**required**): dose rate in Gy/s and dose rate error in Gy/s
method	character (with default): method used for error calculation (gaussian or absolute), see details for further information

Details

Calculation of De values from seconds (s) to gray (Gy)

$$De[Gy] = De[s] * DoseRate[Gy/s])$$

Provided calculation methods for error calculation: **gaussian** error propagation

$$De.error.gray = \sqrt{(dose.rate * De.error.seconds)^2 + (De.seconds * dose.rate.error)^2))}$$

absolute error propagation

$$De.error.gray = abs(dose.rate * De.error.seconds) + abs(De.seconds * dose.rate.error)$$

Value

Returns a data.frame with converted values.

Function version

0.3 (2014-04-13 14:32:24)

Note

If no or a wrong method is given, the execution of the function is stopped.

Author(s)

Sebastian Kreutzer, JLU Giessen (Germany), Michael Dietze, GFZ Potsdam (Germany), Margret C. Fuchs, AWI Potsdam (Germany),
R Luminescence Package Team

References

#

See Also

#

Examples

```
##(1) for dose taken from the example data help file
data(ExampleData.DeValues, envir = environment())
Second2Gray(ExampleData.DeValues, c(0.0438,0.0019))
```

---

sTeve *sTeve - sophisticated Tool for efficient data validation and evaluation*

Description

This function provides a sophisticated routine for comprehensive luminescence dating data analysis.

Usage

```
sTeve (n_frames = 10, t_animation = 2, n.tree = 7, type)
```

Arguments

`n_frames`	`integer` (with default): n frames
`t_animation`	`integer` (with default): t animation
`n.tree`	`integer` (with default): How many trees do you want to cut?
`type`	`integer` (optional): Make a decision: 1, 2 or 3

Details

This amazing sophisticated function validates your data seriously.

Value

Validates your data.

Note

This function should not be taken too seriously.

Author(s)

R Luminescence Team, 2012-2013

References

#

See Also

plot_KDE

Examples

```
##no example available
```

`writeR2BIN` *Export Risoe.BINfileData into Risoe BIN-file*

Description

Exports a Risoe.BINfileData object in a *.bin or *.binx file that can be opened by the Analyst software or other Risoe software.

Usage

```
writeR2BIN(object, file, version, txtProgressBar = TRUE)
```

Arguments

`object`

`file`

`version`

`txtProgressBar`

Details

The structure of the exported binary data follows the data structure published in the Appendices of the Analyst manual p. 42.

If `LTYPE`, `DTYPE` and `LIGHTSOURCE` are not of type `character`, no transformation into numeric values is done.

Value

Write a binary file.

Function version

0.1 (2014-04-11 18:11:51)

Note

The function just roughly checks the data structures. The validity of the output data depends on the user.

The validity of the file path is not further checked.
BIN-file conversions using the argument `version` may be a lossy conversion, depending on the chosen input and output data (e.g. conversion from version 06 to 04).

Author(s)

Sebastian Kreutzer, Freiberg Instruments/JLU Giessen (Germany),
R Luminescence Package Team

References

Duller, G., 2007. Analyst.

See Also

`readBIN2R`, `Risoe.BINfileData`, `writeBin`

Examples

```
##uncomment for usage

##data(ExampleData.BINfileData, envir = environment())
##writeR2BIN(CWOSL.SAR.Data, file="[your path]/output.bin")
```

Index