Package: OSLdecomposition (via r-universe)

September 10, 2024

Type Package

Title Signal Component Analysis for Optically Stimulated Luminescence

Version 1.0.2

Date 2023-09-01

Maintainer Dirk Mittelstraß <dirk.mittelstrass@luminescence.de>

Description Function library for the identification and separation of exponentially decaying signal components in continuous-wave optically stimulated luminescence measurements. A special emphasis is laid on luminescence dating with quartz, which is known for systematic errors due to signal components with unequal physical behaviour. Also, this package enables an easy to use signal decomposition of data sets imported and analysed with the R package 'Luminescence'. This includes the optional automatic creation of HTML reports. Further information and tutorials can be found at <https://luminescence.de>.

Depends R (>= 4.3)

License GPL-3

BugReports https://github.com/DirkMittelstrass/OSLdecomposition/issues

Encoding UTF-8

RoxygenNote 7.3.2

Imports Luminescence (>= 0.9.24), methods, utils, stats, tools, DEoptim, minpack.lm, gridExtra, ggplot2, scales, ggpubr, rmarkdown

Suggests kableExtra, knitr

Repository https://dirkmittelstrass.r-universe.dev

RemoteUrl https://github.com/dirkmittelstrass/osldecomposition

RemoteRef HEAD

RemoteSha 5d1da87c72e7ba59fab15ea48d4066b2795aedba

Contents

OSLdecomposition-package . . . 2
decompose_OSLcurve . . . 3
fit_OSLcurve . . . 7
optimise_OSLintervals . . . 11
plot_OSLcurve . . . 13
plot_PhotoCrosssections . . . 16
RLum.OSL_correction . . . 18
RLum.OSL_decomposition . . . 22
RLum.OSL_global_fitting . . . 25
simulate_OSLcomponents . . . 28
sum_OSLcurves . . . 30

Index **32**

`OSLdecomposition-package` *Signal Component Analysis for Optically Stimulated Luminescence*

Description

Function library for the identification and separation of exponentially decaying signal components in continuous-wave optically stimulated luminescence (CW-OSL) measurements. A special emphasis is laid on luminescence dating with quartz, which is known for systematic errors due to CW-OSL signal components with unequal physical behaviour. Also, this package enables an easy to use signal decomposition of CW-OSL data sets imported and analysed with the R package 'Luminescence'. This includes the optional automatic creation of HTML reports.

Details

Project website

- `https://luminescence.de`

Source code repository

- `https://github.com/DirkMittelstrass/OSLdecomposition`

Bug reporting

- `https://github.com/DirkMittelstrass/OSLdecomposition/issues`

This package is part of the RLum.Network

- `https://r-luminescence.org`

Package maintainer

Dirk Mittelstraß, Independent researcher, Dresden (Germany),
<dirk.mittelstrass@luminescence.de>

Citation

Mittelstraß, D., Schmidt, C., Beyer, J., Heitmann, J. and Straessner, A.: R package OSLdecomposition: Automated identification and separation of quartz CW-OSL signal components, *in preparation*.

Funding Dirk Mittelstraß created this package as part of his master thesis and further enhanced and published it as private endeavour. He did not receive any specific grant from funding agencies in the public, commercial or not-for-profit sectors.

Author(s)

Maintainer: Dirk Mittelstraß <dirk.mittelstrass@luminescence.de> (ORCID)

Authors:

- Sebastian Kreutzer (ORCID)
- Christoph Schmidt (ORCID)

Other contributors:

- Jan Beyer (ORCID) [thesis advisor]
- Johannes Heitmann [thesis advisor]
- Arno Straessner (ORCID) [thesis advisor]

See Also

Useful links:

- Report bugs at `https://github.com/DirkMittelstrass/OSLdecomposition/issues`

`decompose_OSLcurve` *Multi-exponential CW-OSL decomposition*

Description

Function for determining the signal component amplitudes of a multi-exponential decay curve if the signal component decay parameters are already given. Thus, this function decomposes CW-OSL curves with known components of unknown intensity.

The function assumes multiple exponentially decaying signal components with first-order kinetics:

$$I(t) = n_1\lambda_1 exp(-\lambda_1 t) + n_2\lambda_2 exp(-\lambda_2 t) + ... + n_K\lambda_K exp(-\lambda_K t)$$

with $I(t)$ the CW-OSL signal, n the signal component intensity, λ the signal component decay constant and K the number of signal components. For the actual decomposition procedure, the integrated version of this formula is used, see Mittelstrass et al. (2021) for details.

Decomposition algorithm

The calculation procedure depends on the function argument `algorithm`. This function includes two different decomposition algorithms: `"det"` for **det**erminant solution and `"nls"` for **n**onlinear **l**east **s**quares estimate

`algorithm = "det"` (default)

The function calculates the CW-OSL component intensities by building an equation system which is then solved by a determinant-based approach (Cramers rule). This purely analytical approach gives the algorithm a solution in all possible cases, even if the measurement consists just of noise or the wrong model is used. There are also no 'false minima' events. The statistical error is calculated by applying the *propagation of uncertainty* method on Cramers rule.

The precision of this algorithm as well as the propagation of eventual systematic errors of the decay rate values, depend on the integration intervals, given by the columns `$t.start`, `$t.end`, `$ch.start` and `$ch.end` of the data.frame used as input for the argument `components`. In principle, these can be chosen freely. Reasonable integration intervals are defined by optimise_OSLintervals. If not defined, the logarithmic mean values between life times (reciprocal decay rate) of subsequent components are used as interval borders.

`algorithm = "nls"`

As alternative algorithm, Levenberg-Marquardt nonlinear regression is available, see minpack.lm::nlsLM for details. The results are identical to that of the `"det"` algorithm in accuracy and precision. But there is the slight chance (< 1 %) of fitting failure when using the `"nls"` algorithm. Also, the statistical errors are underestimated by 20-80 % in most cases. As advantage, the `"nls"` algorithm is less sensitive against systematic errors caused by uncorrected signal background.

`algorithm = "det+nls"`

Both algorithms can be combined. Then, `"det"` provides the startings values and the error estimations for `"nls"` and returns replacement results, in case `"nls"` fails. `"nls"` compensates for potential systematic errors in the fast and medium components intensity values due to uncorrected signal background. However, the background signal will still affect slow component results. The slowest component will be overestimated while the second slowest component will be underestimated. If these components are of particular interest, it is recommended to set `background.fitting = TRUE`

All three methods were tested at 5x10^6 simulated CW-OSL curves by Mittelstrass (2019) for their performance (+++ reliable results in all cases; ++ reliable in >95% of cases: + reliable in most cases):

Characteristics	**det**	**nls**	**det+nls**
Decomposition success rate	100 %	>99 %	100 %
Component intensity accuracy	+++	+++	+++
Accuracy in case of uncorrected background	+	++	++
Error estimation accuracy	+++	+	++

In summary, `algorithm = "det"` is recommended for the most cases. If the signal background level is significant (> 2 % of initial signal) but was not corrected, `algorithm = "det+nls"` is the better choice. Setting `background.fitting = TRUE` is usually not recommended, only in case slow components shall be investigated in measurements with uncorrected background.

Error estimation

In case of algorithm = "det" or "det+nls" the Propagation of Uncertainty method is used to transform signal bin error values (column $bin.error) into component intensity error values (column $n.error). The signal bin error calculation depends on the argument error.estimation, see below. If algorithm = "nls" is used, the error values provided by minpack.lm::nlsLM are returned.

error.estimation = "empiric" (default)

The standard deviation of each signal bin (signal bin = signal value of an integrated time interval) is calculated from the *corrected sample variance* between the CW-OSL model and the actual CW-OSL curve for that interval. Thus, statistical errors are monitored accurately without any prior knowledge required. However, potential systematic errors are monitored insufficiently. Also, at least two (better more) data points per signal bin are needed to estimate its standard deviation. If a signal bin consists just of one data point, its square root value is taken as standard deviation, in accordance to the Poisson distribution.

error.estimation = "poisson" or numeric value

Alternatively the standard error can be calculated by approximating a **Poisson** distributed signal error, known as *Shot noise*. This is suitable if the lack of data points on the x-axis circumvents an empiric error estimation, like with spatially or spectrally resolved CCD measurements. Also the parameter can be set to a numeric value, which represents the detector noise in *cts / s* and is assumed to be normal distributed. The detector noise will be added on top of the Poisson distributed shot noise.

error.estimation = "only.bin.RSS"

The error estimation is omitted but the residual sum of squares (RSS) between input curve and combined signal component curves is calculated. However, the RSS value is divided into sections according to the signal bins (column $bin.RSS). The full RSS value can be calculated by summing over the complete column. The RSS value is usually used a minimization target in fitting algorithms, like done in fit_OSLcurve. The values of the $bin.RSS column allows for weighted fitting by applying pre-factors to the bin RSS values. For further speed advance, the calculation of $components$n.residual and $components$initial.signal is also omitted.

error.estimation = "none"

The error estimation is omitted. This option saves significant computing time, if the error estimation is not of significance. For further speed advance, the calculation of $components$n.residual and $components$initial.signal is also omitted.

Systematic errors

The ratio of the error values of both error estimation methods can be used to detect (but not quantify) systematic errors. "poisson" error values are not affected by systematic errors, while "empiric" errors are. If the detector noise is known and taken into account, the relation between both values for a given signal bin should be about $empiric/poisson = 1$. In case of systematic errors, this ratio increases.

Usage

```
decompose_OSLcurve(
  curve,
  components,
  background.fitting = FALSE,
  algorithm = "det",
  error.estimation = "empiric",
```

```
  verbose = TRUE
)
```

Arguments

- `curve` — data.frame or matrix or Luminescence::RLum.Data.Curve (**required**): CW-OSL curve x-Axis: `$time` or first column as measurement time (must have constant time intervals); y-Axis: `$signal` or second column as luminescence signal. Further columns will be ignored.
- `components` — data.frame or numeric vector (**required**): Either a vector containing the decay parameters of the CW-OSL components or a table (data.frame), usually the table returned by fit_OSLcurve. In case of a vector: It is recommended to use less than 7 parameters. The parameters will be sorted in decreasing order. In case of a data.frame, one column must be named `$lambda`. It is recommended to provide also integration interval parameters (columns `$t.start`, `$t.end`, `$ch.start`, `$ch.end`), which can be found by applying optimise_OSLintervals to the global mean curve, calculated by sum_OSLcurves. If one or more column is missing, a simple interval definition algorithm is run automatically, see section **Details**.
- `background.fitting` — logical (*with default*): if `TRUE`, an additional signal component with a decay rate of $\lambda = 0$ is included. This allows for an accurate estimation of slow component intensities if the data is not background corrected. However, the additional component reduces the overall precision of the algorithm. It can also cause implausible slow component results if the measurement duration is not sufficiently long (> 30 s).
- `algorithm` — character string (*with default*): Choice of curve decomposition approach. Either `"det"` or `"det+nls"` or `"nls"`, see section **Details**. ^^
- `error.estimation` — character string (*with default*): integral error estimation approach, either `"empiric"` or `"poisson"` or a numeric value or `"none"`, see section **Details**. This argument has no effect if `algorithm = "nls"`.
- `verbose` — logical (*with default*): enables console text output

Value

The input table **components** data.frame will be returned with added or overwritten columns: `$n`, `$n.error`, `$n.residual`, `$bin`, `$bin.error`, `$bin.RSS`, `$initial.signal`. Which columns are written depends on the selected parameters. If an input data.frame contains already one of the above columns but parameters are selected which do not re-calculate the values, the values of the columns are set to `NA`.

Last updates

2022-07-25, DM: Extended algorithm for bin-wise RSS calculation and added error estimation option "only.bin.RSS"

Author(s)

Dirk Mittelstraß, <dirk.mittelstrass@luminescence.de>

Please cite the package the following way:

Mittelstraß, D., Schmidt, C., Beyer, J., Heitmann, J. and Straessner, A.: R package OSLdecomposition: Automated identification and separation of quartz CW-OSL signal components, *in preparation*.

References

Mittelstraß, D., 2019. Decomposition of weak optically stimulated luminescence signals and its application in retrospective dosimetry at quartz (Master thesis). TU Dresden, Dresden.

See Also

fit_OSLcurve, optimise_OSLintervals, RLum.OSL_decomposition, minpack.lm::nlsLM

Examples

```
# Set some arbitrary decay parameter for a dim CW-OSL measurement of quartz
components <- data.frame(name = c("fast", "medium", "slow"),
                         lambda = c(2, 0.5, 0.02),
                         n = c(1000, 1000, 10000))

# Simulate the CW-OSL curve and add some signal noise and some detection background
curve <- simulate_OSLcomponents(components, simulate.curve = TRUE,
                                add.poisson.noise = TRUE, add.background = 40)

# Decompose the simulated curve
components <- decompose_OSLcurve(curve, components)

# Display the component separation results
plot_OSLcurve(curve, components)

### Decomposition including signal background fitting:

# Define optimized integration intervals, including an interval for the background
components <- optimise_OSLintervals(components, curve, background.component = TRUE)

# Decompose again and view results
components <- decompose_OSLcurve(curve, components, background.fitting = TRUE)
plot_OSLcurve(curve, components)
```

`fit_OSLcurve` *Multi-exponential CW-OSL curve fitting*

Description

Fitting function for multi-exponentially decaying CW-OSL measurements, based on the algorithm described by Bluszcz & Adamiec (2006).

The function assumes multiple exponentially decaying signal components with first-order kinetics:

$$I(t) = n_1\lambda_1 exp(-\lambda_1 t) + n_2\lambda_2 exp(-\lambda_2 t) + ... + n_K\lambda_K exp(-\lambda_K t)$$

with $I(t)$ the CW-OSL signal, n the signal component intensity, λ the signal component decay constant and K the number of signal components. For actual fitting, the integrated version of this formula is used, see Mittelstraß et al. (2021) for details.

The fitting algorithm is an implementation of the *hybrid evolutionary-linear algorithm* (HELA) by Bluszcz & Adamiec (2006). See there or Mittelstraß et al. (in preparation) for details. differential evolution part of HELA is performed by DEoptim::DEoptim. The linear regression part of HELA is performed by decompose_OSLcurve. The parameter refinement by Levenberg-Marquardt fitting is performed by minpack.lm::nlsLM.

F-test

Bluszcz & Adamiec (2006) suggest the use of an F-test to determine the correct number of signal components. This function compares the residual square sum (*RSS_K*) value of each fitting with the RSS_{K-1} value of the previous fitting and calculates an *Improvement-in-fitting-quality* criterion:

$$F_K = (RSS_{K-1} - RSS_K)/2/RSS_K(N - 2K)$$

Here, *N* is the number data points (channels) of the measurement and *K* is the number of OSL components in the fitting model. If *F_K* falls below the threshold value (`F.threshold`), the fitting model with *K* components is apparently not significantly better than the *K* - 1 model of the previous fitting cycle. Thus, the *K* - 1 model will be recommended as fitting solution.

Photoionisation cross sections

While the function is suited for the analysis of a wide variety of multi-exponential decay problems, it is targeted to CW-OSL measurements of quartz under SAR protocol conditions (470 nm stimulation at 125 °C). To compare the calculated OSL components with OSL components reported in published literature, photoionisation cross sections are calculated using the `stimulation.wavelength` λ_{stim} and `stimulation.intensity` Φ_{stim}:

$$\sigma_k = \lambda_k hc/\Phi_{stim}\lambda_{stim}$$

Here σ_k is the photoionisation cross section of component *k* in cm^2, λ_k the CW-OSL decay constant in s^-1, *h* the Planck constant and *c* the speed of light.

If a `stimulation.intensity` between 460 nm and 485 nm is defined, the components are named automatically in accordance to the cross-sections published by Durcan and Duller (2011), Jain et al. (2003) and Singarayer and Bailey (2003). For the Ultrafast and the Slow4 component, no consistent literature values could be found, so their range is tentatively assigned:

Component	Lower limit (cm^2)	Upper limit (cm^2)
Ultrafast	1e-16	1e-15
Fast	1.9e-17	3.1e-17

Medium	3e-18	9e-18
Slow1	1e-18	1.85e-18
Slow2	1.1e-19	4e-19
Slow3	1e-20	4.67e-20
Slow4	1e-21	1e-20

Usage

```
fit_OSLcurve(
  curve,
  K.max = 5,
  F.threshold = 150,
  stimulation.intensity = 30,
  stimulation.wavelength = 470,
  verbose = TRUE,
  output.complex = FALSE,
  parallel.computing = FALSE
)
```

Arguments

`curve` Luminescence::RLum.Data.Curve or data.frame or matrix (**required**): CW-OSL record or average CW-OSL curve created by sum_OSLcurves. If no column `$time` exists, the first column is defined as measurement time (x-axis). Time intervals must be constant. If no column `$signal` exists, the second column is defined as signal values (y-axis). Further columns will be ignored

`K.max` numeric (*with default*): Maximum number of components K. The computing time increases exponentially with the component number. $K < 7$ is recommended

`F.threshold` numeric (*with default*): Fitting stop criterion. If the F-value is lower than this threshold, the fitting procedure stops and the K - 1 fit is returned

`stimulation.intensity` numeric (*with default*): Intensity of optical stimulation in *mW / cm²*. Used to calculate photoionisation cross sections.

`stimulation.wavelength` numeric (*with default*): Wavelength of optical stimulation in *nm*. Used to calculate photoionisation cross sections. If a wavelength between 465 and 480 nm is chosen, the cross sections are set into relation with literature values to name the signal components automatically.

`verbose` logical (*with default*): Enables console text output.

`output.complex` logical (*with default*): If `TRUE`, the function returns a list of objects, see section **Value** for further information. If `FALSE`, the function returns a data.frame with the CW-OSL model parameters of the fitting chosen by the F-test. Setting the parameter to `FALSE` is not recommended when fitting a global average curve created by sum_OSLcurves as over-fitting is likely in such cases.

parallel.computing
: logical (*with default*): Enables the use of multiple CPU cores. This increases the execution speed significantly but may need administrator rights and/or a firewall exception. See DEoptim::DEoptim.control for further information.

Value

If `output.complex = FALSE`, a data.frame is returned. It contains the signal decay rates and signal intensities of the best fit. The best fit was either chosen by the F-test or the last successful fitting iteration.

If `output.complex = TRUE`, a list of objects is returned:

Element	Type	Description
decay.rates	numeric	vector of the best suiting decay rates
K.selected	numeric	number of components of the best fit
F.test	data.frame	table containing the F-test parameter and the decay rates of each fitting model
F.test.print	data.frame	the same table as above, but formated for pretty console and report output
info.text	character	collected messages from the algorithms
component.tables	list	result data.frames for all tested models
curve	data.frame	fitted time-signal-curve
components	data.frame	best fit; same object as `output.complex = FALSE` returns
fit.results	list	list of nls objects for all tested models
plot.data	data.frame	factorized results for overview plotting with plot_PhotoCrosssections
parameters	list	function arguments and the needed computing time

Last update

2022-07-27, DM: Moved residual sum of squares (RSS) calculation during DE-optimization cycle to decompose_OSLcurve() to improve computing time by factor 3 to 4

Author(s)

Dirk Mittelstraß, <dirk.mittelstrass@luminescence.de>

Please cite the package the following way:

Mittelstraß, D., Schmidt, C., Beyer, J., Heitmann, J. and Straessner, A.: R package OSLdecomposition: Automated identification and separation of quartz CW-OSL signal components, *in preparation*.

References

Bluszcz, A., Adamiec, G., 2006. Application of differential evolution to fitting OSL decay curves. Radiation Measurements 41, 886–891.

Durcan, J.A., Duller, G.A.T., 2011. The fast ratio: A rapid measure for testing the dominance of the fast component in the initial OSL signal from quartz. Radiation Measurements 46, 1065–1072.

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

Mittelstraß, D., 2019. Decomposition of weak optically stimulated luminescence signals and its application in retrospective dosimetry at quartz (Master thesis). TU Dresden, Dresden.

Singarayer, J.S., Bailey, R.M., 2003. Further investigations of the quartz optically stimulated luminescence components using linear modulation. Radiation Measurements, Proceedings of the 10th international Conference on Luminescence and Electron-Spin Resonance Dating (LED 2002) 37, 451–458.

See Also

RLum.OSL_decomposition, sum_OSLcurves, decompose_OSLcurve, plot_OSLcurve, plot_PhotoCrosssections, minpack.lm::nlsLM, DEoptim::DEoptim

Examples

```
# Create a simple curve with just one component
curve <- data.frame(
  X = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12),
  Y = c(377, 244, 163, 93, 59, 28, 17, 13, 10, 8, 9, 5))
# Perform fitting
components <- fit_OSLcurve(curve, F.threshold = 3)

# Display results
plot_OSLcurve(curve, components)
```

---

optimise_OSLintervals *Find adequate integration intervals for CW-OSL decomposition*

Description

This function defines integration intervals for CW-OSL component separation with decompose_OSLcurve. The underlying iterative optimisation process aims for minimum cross-correlation between the signal components.

The precision of the component separation with decompose_OSLcurve and the impact of systematic decay rate errors on the component separation depends on the integration interval definition. This function minimises the influence of an under/over-estimated decay rate to the signal intensity calculation of other component. This is done by maximizing the denominator determinant in Cramers rule, see Mittelstraß (2019) for details. For maximisation, the iterative evolutionary algorithm of Storn and Price (1997) is used, available in *R* through DEoptim::DEoptim.

The inclusion of a background component is supported, see decompose_OSLcurve for details.

Usage

```
optimise_OSLintervals(
  components,
  curve = NULL,
  channel.width = NA,
```

```
  channel.number = NA,
  t.start = 0,
  t.end = NA,
  background.component = FALSE,
  verbose = TRUE,
  parallel.computing = FALSE
)
```

Arguments

components — data.frame or numeric vector (**required**): Table or vector containing the decay constants of the signal components. A data.frame must contain a column `$lambda`. Usually the data.frame is provided by fit_OSLcurve.

curve — data.frame or matrix or Luminescence::RLum.Data.Curve (*optional*): OSL signal curve which serves as time axis template. The input curve will be used to define `channel.width` and `channel.number`

channel.width — numeric (*optional*): Channel width in seconds. Necessary if `curve` is not given.

channel.number — numeric (*optional*): Number of channels resp. data points. Necessary if `curve` is not given.

t.start — numeric (*with default*): Starting time of the first interval, per default the start of the measurement.

t.end — numeric (*optional*): End time of the last interval, per default the end of the measurement.

background.component — logical (*with default*): If `TRUE`, an additional interval for a component with a decay rate of zero will be determined. This enables the calculation of the signal background level during the signal decomposition with decompose_OSLcurve.

verbose — logical (*with default*): Enables console text output.

parallel.computing — logical (*with default*): Enables the use of multiple CPU cores. This increases the execution speed significantly but may need administrator rights and/or a firewall exception. See DEoptim::DEoptim.control for further information.

Value

The input table `components` data.frame will be returned with four additional columns: `$t.start`, `$t.end` defining the time intervals and `$ch.start`, `$ch.end` assigning those intervals to channel indicies. If a numeric vector is given as input, a new data.frame will be returned.

Last updates

2020-08-23, DM: Replaced previous maximum searching algorithm with DEoptim::DEoptim (**update may have changed analysis results**)

2020-10-29, DM: Added `parallel.computing` argument; enhanced roxygen documentation (*minor update*)

Author(s)

Dirk Mittelstraß, <dirk.mittelstrass@luminescence.de>

Please cite the package the following way:

Mittelstraß, D., Schmidt, C., Beyer, J., Heitmann, J. and Straessner, A.: R package OSLdecomposition: Automated identification and separation of quartz CW-OSL signal components, *in preparation*.

References

Mittelstraß, D., 2019. Decomposition of weak optically stimulated luminescence signals and its application in retrospective dosimetry at quartz (Master thesis). TU Dresden, Dresden.

Storn, R., Price, K., 1997. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization 11, 341–359.

See Also

decompose_OSLcurve, RLum.OSL_decomposition, DEoptim::DEoptim, fit_OSLcurve

Examples

```
A <- optimise_OSLintervals(c(2, 0.5, 0.02), channel.width = 0.1, channel.number = 200)
print(A, row.names = FALSE)
```

`plot_OSLcurve` *Advanced plot function for component resolved CW-OSL curves*

Description

This function is used for plotting CW-OSL curves and its signal components. It can handle data returned by fit_OSLcurve or decompose_OSLcurve. Besides CW-OSL curves, pseudoLM-OSL curves and residual plots can also be plotted.

Change graph types with parameter: `display`

`"detailed"`	(default) Output plot consists of: Linear CW-OSL plot, pseudoLM-OSL plot, residual curve and componen
`"lin"`	Linear CW-OSL plot only
`"compare_lin"`	Linear CW-OSL plot with residual curve below and component table on bottom. Useful if two CW-OSL m
`"log"`	CW-OSL plot with logarithmic y-Axis and linear x-Axis
`"compare_log"`	CW-OSL plot with logarithmic y-Axis with residual curve below and component table on bottom. Useful i
`"loglog"`	Double-logarithmic CW-OSL plot
`"LM"`	PseudoLM-OSL plot
`"res"`	Plot of residual curve: Measurement minus fitting model
`"tab"`	Table of component parameters as image
`"raw"`	Raw x-y plot without further data

PseudoLM-OSL curves are created using the transformation described by Bulur (2000). The stimulation ramp duration is twice the CW-OSL duration. See Bos and Wallinga (2012) for a detailed explanation and discussion.

Usage

```
plot_OSLcurve(
  curve = NULL,
  components = NULL,
  display = "detailed",
  show.legend = TRUE,
  show.intervals = FALSE,
  show.crosssec = FALSE,
  show.initial = FALSE,
  theme.set = ggplot2::theme_classic(),
  title = NULL,
  hide.plot = FALSE,
  filename = NULL
)
```

Arguments

`curve`	data.frame or matrix or Luminescence::RLum.Data.Curve (*optional*): CW-OSL curve x-Axis: `$time` or first column as measurement time (must have constant time intervals); y-Axis: `$signal` or second column as luminescence signal. Other columns will be plotted as component curves, in case no input object `components` is defined. If no input is given, a CW-OSL curve will be simulated with the parameters of `components`
`components`	data.frame (*optional*): Table with OSL component parameters. The parameters are used to approximate separate signal decay curves for each component. Need to have at least the columns: `$names`, `$lambda` and `$n`. If an insufficient or no input object is provided, the 'curve" object will be searched for component-related signal values.
`display`	character (*with default*): Sets the arrangement of graphs, see section **Details**.
`show.legend`	logical (*with default*): Draws a legend in the top right corner of the first graph.
`show.intervals`	logical (*with default*): Draws vertical lines into the residual plot showing the signal bin intervals (if available) for the CW-OSL decomposition with decompose_OSLcurve.
`show.crosssec`	logical (*with default*): Displays photoionisation cross section values in the component table (if available).
`show.initial`	logical (*with default*): Displays signal share at the first channel in the component table (if available).
`theme.set`	ggplot2::ggplot2-package object (*with default*): Graphical theme of the output plot. This argument is forwarded to ggplot2::theme_set. Recommended themes are `ggplot2::theme_minimal()`, `ggplot2::theme_classic()` and `ggplot2::theme_bw()`, see ggplot2::theme_bw or here for a full list.
`title`	character (*with default*): Plot title. Overwrites automatic titles but affects just the first (upper left) graph in case of multi-graph display setting. Set `title = NULL` for auto-title and `title = ""` for no title.

hide.plot — logical (*with default*): If true, plot is not drawn but can still be saved as file or caught by `A <- plot_OSLcurve(...)`. If caught, the plot can be drawn manually for example by using gridExtra::grid.arrange.

filename — character (*optional*): File name or path to save the plot as image. If just a file name is given, the image is saved in the working directory. The image type is chosen by the file ending. Both, vector images as well as pixel images are possible. Allowed are `.pdf`, `.eps`, `.svg` (vector graphics), `.jpg`, `.png`, `.bmp` (pixel graphics) and more, see ggplot2::ggsave.

Value

An invisible ggplot2::ggplot object containing the diagram will returned. "Invisible" means, the no value will be returned (e.g. no console printout) if the function is not assigned to a variable via <-. If the function is assigned, the returned object can be further manipulated by ggplot2::ggplot2-package methods or manually drawn by various functions like for example gridExtra::grid.arrange.

Last update

2021-03-29, DM: Hidden output objects are now ggplot2::ggplot2-package objects if the plot is not a composite diagram

Author(s)

Dirk Mittelstraß, <dirk.mittelstrass@luminescence.de>

Please cite the package the following way:

Mittelstraß, D., Schmidt, C., Beyer, J., Heitmann, J. and Straessner, A.: R package OSLdecomposition: Automated identification and separation of quartz CW-OSL signal components, *in preparation*.

References

Bos, A. J. J. and Wallinga, J., 2012. How to visualize quartz OSL signal components, Radiation Measurements, 47(9)

Bulur, E., 2000. A simple transformation for converting CW-OSL curves to LM-OSL curves, Radiation Measurements, 32(2)

See Also

fit_OSLcurve, [RLum.OSL_decomposition, RLum.OSL_global_fitting, simulate_OSLcomponents

Examples

```
# Set some arbitrary decay parameter for a dim CW-OSL measurement of quartz
components <- data.frame(name = c("fast", "medium", "slow"),
                         lambda = c(2, 0.5, 0.02),
                         n = c(1000, 1000, 10000))

# Simulate a CW-OSL curve including some signal noise
curve <- simulate_OSLcomponents(components, simulate.curve = TRUE, add.poisson.noise = TRUE)
```

```
# Display the simulated curve
plot_OSLcurve(curve, components)
```

---

plot_PhotoCrosssections

Plot comparison of CW-OSL component photoionisation cross sections of different models

---

Description

This function takes the `output.complex = TRUE` output of fit_OSLcurve and draws the photoionisation cross sections of different models in relation to each other. If a stimulation wavelength between 465 and 480 nm was chosen, the photoionisation cross sections are also set in relation to literature values from Singarayer and Bailey (2003), Jain et al. (2003) and Durcan and Duller (2011).

The photoionisation cross section ranges of the reference components are defined as following:

Component	**Lower limit (cm^2)**	**Upper limit (cm^2)**
Ultrafast	1e-16	1e-15
Fast	1.9e-17	3.1e-17
Medium	3e-18	9e-18
Slow1	1e-18	1.85e-18
Slow2	1.1e-19	4e-19
Slow3	1e-20	4.67e-20
Slow4	1e-21	1e-20

Usage

```
plot_PhotoCrosssections(
  fit.list,
  stimulation.intensity = NULL,
  stimulation.wavelength = NULL,
  K.selected = NULL,
  title = NULL,
  hide.plot = FALSE,
  filename = NULL
)
```

Arguments

`fit.list` list (**required**): Output object of fit_OSLcurve. The object must be created with the setting `output.complex = TRUE`.

`stimulation.intensity`
numeric (*optional*): Intensity of optical stimulation in *mW / cm²*. Used to calculate the photoionisation cross sections. If not given, the input value for fit_OSLcurve is used

stimulation.wavelength
: numeric (*optional*): Wavelength of optical stimulation in nm. Used to calculate the photoionisation cross sections. If not given, the input value for fit_OSLcurve is used

K.selected
: numeric (*optional*): Draws a red rectangle around the `K = K.selected` row, thus highlighting the model of choice.

title
: character (*with default*): Plot title. Set `title = NULL` for no title.

hide.plot
: logical (*with default*): If true, plot is not drawn but can still be saved as file or caught by `A <- plot_PhotoCrosssections(...)`. If caught, the plot can be drawn manually for example by using gridExtra::grid.arrange.

filename
: character (*optional*): File name or path to save the plot as image. If just a file name is given, the image is saved in the working directory. The image type is chosen by the file ending. Both, vector images as well as pixel images are possible. Allowed are `.pdf`, `.eps`, `.svg` (vector graphics), `.jpg`, `.png`, `.bmp` (pixel graphics) and more, see ggplot2::ggsave.

Value

An invisible ggplot2::ggplot object containing the diagram will returned. "Invisible" means, the no value will be returned (e.g. no console printout) if the function is not assigned to a variable via `<-`. If the function is assigned, the returned object can be further manipulated with ggplot2::ggplot2-package methods or manually drawn by various functions like for example gridExtra::grid.arrange.

Last updates

2020-11-04, DM: Added roxygen documentation

Author(s)

Dirk Mittelstraß, `<dirk.mittelstrass@luminescence.de>`

Please cite the package the following way:

Mittelstraß, D., Schmidt, C., Beyer, J., Heitmann, J. and Straessner, A.: R package OSLdecomposition: Automated identification and separation of quartz CW-OSL signal components, *in preparation*.

References

Durcan, J.A., Duller, G.A.T., 2011. The fast ratio: A rapid measure for testing the dominance of the fast component in the initial OSL signal from quartz. Radiation Measurements 46, 1065–1072.

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

Singarayer, J.S., Bailey, R.M., 2003. Further investigations of the quartz optically stimulated luminescence components using linear modulation. Radiation Measurements, Proceedings of the 10th international Conference on Luminescence and Electron-Spin Resonance Dating (LED 2002) 37, 451–458.

See Also

fit_OSLcurve, RLum.OSL_global_fitting

Examples

```
# Set some arbitrary decay parameter for a dim CW-OSL measurement of quartz
name <- c("fast", "slow")
lambda <- c(2, 0.02)
n <- c(1e6, 5e7)

# Build a component table
components <- data.frame(name, lambda, n)

# Simulate the CW-OSL curve and add some signal noise
curve <- simulate_OSLcomponents(components, simulate.curve = TRUE, add.poisson.noise = TRUE)

# Perform nonlinear regression at the simulated curve
fit_results <- fit_OSLcurve(curve, K.max = 2, output.complex = TRUE)

# Plot the fitting iterations and set them into context
plot_PhotoCrosssections(fit_results)
```

`RLum.OSL_correction` *Check and correct CW-OSL curves in RLum.Analysis data sets*

Description

CW-OSL measurements are often affected by background signals or might be measured under inconsistent detection settings. This function provides tools to test and solve some common problems.

This function processes data sets created within the Luminescence::Luminescence-package (Kreutzer et al. 2012). Those data sets must be formatted as Luminescence::RLum.Analysis objects. Output objects will also be Luminescence::RLum.Analysis objects and are meant for further analysis with RLum.OSL_global_fitting.

The data preparation tools are executed in the following order:

1. `check_consistency`
2. `remove_light_off`
3. `limit_duration`
4. `PMT_pulse_pair_resolution`
5. `background_sequence`
6. `subtract_offset`

Currently, not all functions are available.

Details to `remove_light_off`: The algorithm does the following: (1) Create global reference curve with sum_OSLcurves (2) Search for the maximum in the first half of the reference curve and remove all data points before the maximum . Do this for all curves of the selected 'record_type'. (3) Search for an infliction point with negative curvature (minimum of second differential) in the second half of the reference curve. If the next data point has at least 50% less signal, remove all data points after the infliction point. Do this for all curves of the selected 'record_type'.

Details to `PMT_pulse_pair_resolution`:

The algorithm corrects non-linearity of signal values due to insufficient pulse-pair resolution of the photo-multiplier tube (PMT). Equation (6-2) of the *Hamamatsu Photomultiplier Handbook* is used:

$$I_corrected = I_measured/(1 - I_measured * resolution)$$

The algorithm does not account for PMT saturation and PMT aging effects. As default pulse-pair resolution 18 *ns* is pre-defined, the *Hamamatsu* H7360 series pulse-pair resolution according to the data sheet. The H7360-02 is the default PMT in *Freiberg Instruments lexsyg* OSL/TL readers. *DTU Physics Risoe* TL/OSL reader deploy *ET Enterprise* 9235B series PMTs as default. For these PMTs, the pulse-pair resolutions is not given in the data sheets and relies on the operation voltage. However, due to the pulse properties given in the data sheets, it is unlikely that those PMTs have a better pulse-pair resolution than 18 *ns*.

Impact of a pulse-pair resolution correction of 18 ns

Measured signal	Corrected signal	Signal underestimation
1000 cts/s	1000 cts/s	0.00 %
10000 cts/s	10002 cts/s	0.02 %
50000 cts/s	50045 cts/s	0.09 %
100000 cts/s	100180 cts/s	0.18 %
500000 cts/s	504541 cts/s	0.91 %
1000000 cts/s	1018330 cts/s	1.83 %

Usage

```
RLum.OSL_correction(
  object,
  record_type = "OSL",
  background_sequence = NULL,
  subtract_offset = 0,
  check_consistency = TRUE,
  remove_light_off = TRUE,
  limit_duration = 20,
  PMT_pulse_pair_resolution = 18,
  verbose = TRUE
)
```

Arguments

object
: Luminescence::RLum.Analysis or list of Luminescence::RLum.Analysis (**required**): Data set of one or multiple CW-OSL measured aliquots.

record_type
: character (*with default*): Type of records selected from the input `object`, see `object[[]]@records[[]]@recordType`. Common are: `"OSL"`,`"SGOSL"` or `"IRSL"`.

background_sequence
: numeric vector (*optional*): Indices of list items with CW-OSL measurements of empty aliquots. The records in these list items are used to calculate one average CW-OSL background curve with sum_OSLcurves. This background curve is subtracted from each CW-OSL record of the data set. The attributes `@recordType` of the background measurements will be renamed to `"{record_type}background"`.

subtract_offset
: numeric (*optional*): Signal offset value in counts per second (*cts/s*). Value is handled as background level and will be subtracted from each CW-OSL record.

check_consistency
: logical (*with default*): The CW-OSL component identification and separation procedure requires uniform detection parameters throughout the whole data set. If `TRUE`, all records are compared for their channel width and their number of channels. Those records with the most frequent set of channel parameters keep their `@recordType` attribute, while records with other sets of measurement parameter will be enumerated `record_type` `"{record_type}2"`, `"{record_type}3"` and so on.

remove_light_off
: logical (*with default*): Checks if the records contain zero-signal intervals at beginning and/or end of the measurement and removes them. Useful to tailor single-grain measurements.

limit_duration
: numeric (*with default*): Reduce measurement duration to input value in seconds (*s*). Long measurement duration can lead to over-fitting at the component identification of Step 1 which may induce systematic errors, see Mittelstrass (2019). Thus, limiting the OSL record length ensures sufficient accuracy regarding the Fast and Medium component analysis. If however, slow decaying components are of interest, `limit_duration = NA` is recommended.

PMT_pulse_pair_resolution
: numeric (*with default*): Time span of the pulse-pair resolution of the PMT in nanoseconds (*ns*). If a value is given, the signal values will be corrected for time-resolution related non-linearity at height counting rates, see *Details*. Set `PMT_pulse_pair_resolution = NA` if algorithm shall be omitted.

verbose
: logical (*with default*): Enables console text output.

Value

The input `object`, a list of Luminescence::RLum.Analysis objects, is given back with eventual changes in the elements `object[[]]@records[[]]@recordType` and `object[[]]@records[[]]@data`.

The returned data set contains a new list element `object[["CORRECTION"]]` which provides a list of the input parameters and additional data depending on the applied tools.

Last updates

2023-09-01, DM: Improved input checks to return more helpful messages

Author(s)

Dirk Mittelstrass, <dirk.mittelstrass@luminescence.de>

Please cite the package the following way:

Mittelstraß, D., Schmidt, C., Beyer, J., Heitmann, J. and Straessner, A.: R package OSLdecomposition: Automated identification and separation of quartz CW-OSL signal components, *in preparation*.

References

Hamamatsu, 2007. Photomultiplier Tubes: Basics and Applications, Third Edition (Edition 3A). Hamamatsu Photonics K. K., Hamamatsu City.

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 (1), 1-8.

Mittelstraß, D., 2019. Decomposition of weak optically stimulated luminescence signals and its application in retrospective dosimetry at quartz (Master thesis). TU Dresden, Dresden.

See Also

RLum.OSL_global_fitting, RLum.OSL_decomposition, sum_OSLcurves

Examples

```
# 'FB_10Gy' is a dose recovery test with the Fontainebleau quartz
# measured with a lexsyg research with green LED stimulation
data_path <- system.file("examples", "FB_10Gy_SAR.bin", package = "OSLdecomposition")
data_set <- Luminescence::read_BIN2R(data_path, fastForward = TRUE)

# To correct for the background signal, subtracted the average curve from the
# OSL curves of an empty aliquot (list item 11) from all other OSL records:
data_set_corrected <- RLum.OSL_correction(data_set, background = 11, remove_light_off = FALSE)


# Plot background corrected global average CW-OSL curve
sum_OSLcurves(data_set_corrected, output.plot = TRUE, record_type = "OSL")

# Plot background curve
sum_OSLcurves(data_set_corrected, output.plot = TRUE, record_type = "OSLbackground")
```

RLum.OSL_decomposition
Separate CW-OSL components in RLum.Analysis data sets

Description

Calculates the CW-OSL signal component intensities for each CW-OSL measurement under the requirement that the decay rates are already given. The signal decomposition process uses an analytical approach described in detail in Mittelstrass (2019) and Mittelstrass et al. (in preparation). This function processes Luminescence::RLum.Analysis data sets created within the Luminescence::Luminescence-package (Kreutzer et al. 2012).

The workflow of this function is as follows:

1. optimise_OSLintervals: Approximates the optimal integration intervals. Uses the global average curve as time axis template. If none global average curve is given, one is automatically created using sum_OSLcurves.
2. decompose_OSLcurve: Calculates component intensities for **all** `record_type` measurements. Uses the `"det"` algorithm if a background correction was performed with RLum.OSL_correction or the `"det+nls"` algorithm if no background correction was performed. For error estimation, the `"empiric"` approach is used.
3. Creates a `html` report to summarize the results (*optional*).

Data sets must be formatted as Luminescence::RLum.Analysis objects and should have been processed with RLum.OSL_correction and RLum.OSL_global_fitting beforehand. Output objects are also Luminescence::RLum.Analysis objects and are meant for equivalent dose determination with Luminescence::analyse_SAR.CWOSL.

If `report = TRUE`, a `html` report of the results is rendered by the rmarkdown::rmarkdown-package and saved in the working directory, which is usually the directory of the data file. This report can be displayed, shared and published online without any requirements regarding the operation system or installed software. However, an internet connection is needed to display the *MathJax* encoded equations and special characters. The *Rmarkdown* source code of the report can be found with the following command:

`system.file("rmd", "report_Step2.Rmd", package = "OSLdecomposition")`

Usage

```
RLum.OSL_decomposition(
  object,
  record_type = "OSL",
  K = 3,
  decay_rates = NULL,
  report = FALSE,
  report_dir = NULL,
  image_format = "pdf",
  open_report = TRUE,
```

```
  rmd_path = NULL,
  verbose = TRUE
)
```

Arguments

object	Luminescence::RLum.Analysis or list of Luminescence::RLum.Analysis (**required**): Data set of one or multiple CW-OSL measured aliquots. The data set must either contain a list element `$OSL_COMPONENTS` or the parameter `decay_rates` must be defined.
record_type	character (*with default*): Type of records, selected by the Luminescence::RLum.Analysis attribute `@recordType`. Common are: `"OSL"`,`"SGOSL"` or `"IRSL"`.
K	numeric (*with default*): Number of components. Selects the according result table in the `$OSL_COMPONENTS` list item of the data set `object`.
decay_rates	numeric vector or data.frame (*optional*): User-defined component decay rates. If this parameter is defined, the parameter `K` will ignored. If the input object is a data.frame, then the decay rates must be stored in the column `$lambda`.
report	logical (*with default*): Creates a `html` report, saves it in the `report_dir` directory. The report contains the results and detailed information on the data processing.
report_dir	character (*optional*): Path of output directory if `report = TRUE`. If `report_dir = NULL` (default), a temporary folder is used which is deleted when the R session is closed. File paths are also allowed as parameter, then a new directory named after the OSL data file will be created.
image_format	character (*with default*): Image format of the automatically saved graphs if `report = TRUE` and `report_dir` is set. Allowed are `.pdf`, `.eps`, `.svg` (vector graphics), `.jpg`, `.png`, `.bmp` (pixel graphics) and more, see ggplot2::ggsave. The images are saved in the `report_dir` subfolder `/report_figures`. Set `image_format = NULL` if no images shall be saved.
open_report	logical (*with default*): If set to `TRUE` a browser window displaying the report will be opened automatically.
rmd_path	character (*with default*): **For advanced users:** File path to the rmarkdown::rmarkdown-package source code file of the report. This allows to execute a manipulated version of the report.
verbose	logical (*with default*): Enables console text output.

Value

The input `object`, a list of Luminescence::RLum.Analysis objects is returned but with a new list element `object[["DECOMPOSITION"]]`, containing:

- `$decompositon.input` data.frame: Set of input components. Relevant is just the column `$lambda`
- `$results` data.frame: Overview table of decomposition
- `$parameters` list: Input and algorithm parameters

The Luminescence::RLum.Data.Curve attribute `@info` of each CW-OSL record contains the new entry `$COMPONENTS` with the curve-individual signal component parameters. It can be read for example by:

`object[[i]]@records[[j]]@info[["COMPONENTS"]]`

Last updates

2023-09-01, DM: Improved input checks to return more helpful messages

Author(s)

Dirk Mittelstrass, `<dirk.mittelstrass@luminescence.de>`

Please cite the package the following way:

Mittelstraß, D., Schmidt, C., Beyer, J., Heitmann, J. and Straessner, A.: R package OSLdecomposition: Automated identification and separation of quartz CW-OSL signal components, *in preparation*.

References

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 (1), 1-8.

Mittelstraß, D., 2019. Decomposition of weak optically stimulated luminescence signals and its application in retrospective dosimetry at quartz (Master thesis). TU Dresden, Dresden.

See Also

RLum.OSL_global_fitting, decompose_OSLcurve, optimise_OSLintervals, Luminescence::analyse_SAR.CWOSL

Examples

```
#'FB_10Gy' is a dose recovery test with the Fontainebleau quartz
# measured in a lexsyg research with green LED stimulation
data_path <- system.file("examples", "FB_10Gy_SAR.bin", package = "OSLdecomposition")
data_set <- Luminescence::read_BIN2R(data_path, fastForward = TRUE)

# Separate components
data_set_decomposed <- RLum.OSL_decomposition(
data_set, decay_rates = c(0.8, 0.05))
```

RLum.OSL_global_fitting *Identify CW-OSL signal components in RLum.Analysis data sets*

Description

First, all CW-OSL records are combined to one global average CW-OSL curve, then the multi-exponential fitting approach of Bluszcz and Adamiec (2006) is applied. This function processes Luminescence::RLum.Analysis data sets created within the Luminescence::Luminescence-package (Kreutzer et al. 2012).

The workflow of this function is as follows:

1. sum_OSLcurves: Combine all measurements of type `record_type` to one global average curve.
2. fit_OSLcurve: Identify OSL components by a multi-exponential fitting.
3. Create a `html` report to summarize the results (*optional*).

Data sets must be formatted as Luminescence::RLum.Analysis objects and should have been processed with RLum.OSL_correction beforehand. Output objects are also Luminescence::RLum.Analysis objects and are meant for further analysis with RLum.OSL_decomposition.

If `report = TRUE`, a `html` report of the results is rendered by the rmarkdown::rmarkdown-package and saved in the working directory, which is usually the directory of the data file. This report can be displayed, shared and published online without any requirements to the operation system or installed software. However, an internet connection is needed to display the *MathJax* encoded equations and special characters. The *Rmarkdown* source code of the report can be found with the following command:

`system.file("rmd", "report_Step1.Rmd", package = "OSLdecomposition")`

Usage

```
RLum.OSL_global_fitting(
  object,
  record_type = "OSL",
  K_maximum = 5,
  F_threshold = 150,
  stimulation_intensity = 35,
  stimulation_wavelength = 470,
  report = FALSE,
  report_dir = NULL,
  image_format = "pdf",
  open_report = TRUE,
  rmd_path = NULL,
  verbose = TRUE
)
```

Arguments

`object`	Luminescence::RLum.Analysis or list of Luminescence::RLum.Analysis (**required**): Data set of one or multiple CW-OSL measured aliquots.
`record_type`	character (*with default*): Type of records, selected by the Luminescence::RLum.Analysis attribute `@recordType`. Common are: `"OSL"`,`"SGOSL"` or `"IRSL"`.
`K_maximum`	numeric (*with default*): Maximum number of components *K*, see fit_OSLcurve.
`F_threshold`	numeric (*with default*): Fitting stop criterion, see fit_OSLcurve.
`stimulation_intensity`	numeric (*with default*): Intensity of optical stimulation in *mW / cm²*. Used to calculate photo-ionisation cross-sections, see fit_OSLcurve.
`stimulation_wavelength`	numeric (*with default*): Wavelength of optical stimulation in *nm*. Used to calculate photo-ionisation cross-sections, see fit_OSLcurve.
`report`	logical (*with default*): Creates a `html` report, saves it in the `report_dir` directory. The report contains the results and detailed information on the data processing.
`report_dir`	character (*optional*): Path of output directory if `report = TRUE`. If `report_dir = NULL` (default), a temporary folder is used which is deleted when the R session is closed. File paths are also allowed as parameter, then a new directory named after the OSL data file will be created.
`image_format`	character (*with default*): Image format of the automatically saved graphs if `report = TRUE` and `report_dir` is set. Allowed are `.pdf`, `.eps`, `.svg` (vector graphics), `.jpg`, `.png`, `.bmp` (pixel graphics) and more, see ggplot2::ggsave. The images are saved in the `report_dir` subfolder `/report_figures`. Set `image_format = NULL` if no images shall be saved.
`open_report`	logical (*with default*): If set to `TRUE` a browser window displaying the report will be opened automatically.
`rmd_path`	character (*with default*): **For advanced users:** File path to the rmarkdown::rmarkdown-package source code file of the report. This allows to execute manipulated versions of the report.
`verbose`	logical (*with default*): Enables console text output.

Value

The input `object`, a list of Luminescence::RLum.Analysis objects is returned but with a new list element `object[["OSL_COMPONENTS"]]`, containing:

- `$decay.rates` numeric vector: Decay rates of F-test recommendation or last successful fitting.
- `$K.selected` numeric: Number of components of F-test recommendation or last successful fitting.
- `$F.test` data.frame: F-test table.
- `$F.test.print` data.frame: F-test table but formatted for console output and display with knitr::kable.
- `$info.text` list: Short process log.

- `$component.tables` list of data.frames: Signal component tables for all curve models.
- `$curve` list: Global average curve created from all `record_type` curves in the data set.
- `$components` data.frame: Signal component table of F-test recommendation or last successful fitting.
- `$fit.results` list: Returned fitting objects of DEoptim::DEoptim and minpack.lm::nlsLM for all curve models.
- `$plot.data` data.frame: Model overview table for photo-ionisation cross-section plotting with plot_PhotoCrosssections.
- `$parameters` list: Input and algorithm parameters.

Last updates

2023-09-01, DM: Improved input checks to return more helpful messages

Author(s)

Dirk Mittelstrass, `<dirk.mittelstrass@luminescence.de>`

Please cite the package the following way:

Mittelstraß, D., Schmidt, C., Beyer, J., Heitmann, J. and Straessner, A.: R package OSLdecomposition: Automated identification and separation of quartz CW-OSL signal components, *in preparation*.

References

Bluszcz, A., Adamiec, G., 2006. Application of differential evolution to fitting OSL decay curves. Radiation Measurements 41, 886–891.

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 (1), 1-8.

See Also

RLum.OSL_correction, RLum.OSL_decomposition, sum_OSLcurves, fit_OSLcurve

Examples

```
# 'FB_10Gy' is a dose recovery test with the Fontainebleau quartz
# measured in a lexsyg research with green LED stimulation
data_path <- system.file("examples", "FB_10Gy_SAR.bin", package = "OSLdecomposition")
data_set <- Luminescence::read_BIN2R(data_path, fastForward = TRUE)

# Check data set and perform background correction
data_set_corrected <- RLum.OSL_correction(data_set,
 background = 11,
 remove_light_off = FALSE)

# Identify components
data_set_fitted <- RLum.OSL_global_fitting(
 data_set_corrected,
 K_maximum = 2,
```

```
stimulation_intensity = 50,
stimulation_wavelength = 530)
```

---

simulate_OSLcomponents

Simulates signal component decay curves and whole CW-OSL curves

---

Description

This function builds a bulk CW-OSL curve and CW-OSL component decay curves from OSL component parameters. Therewith it supports fit_OSLcurve, decompose_OSLcurve and plot_OSLcurve by providing model and residual curves.

Usage

```
simulate_OSLcomponents(
  components,
  curve = NULL,
  channel.width = 0.1,
  channel.number = 400,
  simulate.curve = FALSE,
  add.poisson.noise = TRUE,
  add.gaussian.noise = 0,
  add.background = 0,
  round.values = TRUE
)
```

Arguments

`components`	data.frame (**required**): Table with component parameters. The table requires columns `$names`, `$lambda` and `$n`, see section **Examples**.
`curve`	data.frame (*optional*): CW-OSL curve serving as template for the time axis. The input table requires a column `$time`. If no input object is given or the object contains no column `$signal`, `simulate.curve` will be set `TRUE`.
`channel.width`	numeric (*optional*): Channel width in seconds. Necessary for curve simulation if curve is not given.
`channel.number`	numeric (*optional*): Number of channels resp. data points. Necessary for curve simulation if `curve` is not given.
`simulate.curve`	logical (*with default*): Decides if the bulk CW-OSL signal shall be calculated from the component parameter. If `FALSE`, the output curve will take over the column `$signal` from the input `curve`. If `TRUE`, a new column `$signal` will be created which is the sum of all component curves.
`add.poisson.noise`	logical (*with default*): Adds poisson distributed shot noise to `$signal` if `simulate.curve = TRUE`.

add.gaussian.noise
: numeric (*with default*): Standard deviation of the detector noise in *cts/s*, added to `$signal` if `simulate.curve = TRUE`.

add.background
: numeric (*with default*): signal background level in *cts/s*, added to `$signal` if `simulate.curve = TRUE`

round.values
: logical (*with default*): Rounds `$signal` values to integers if `simulate.curve = TRUE`.

Value

A data.frame of a CW-OSL curve with the columns: `$time`, `$signal`, `$residual`, `$sum` and a signal decay curve for each single component named after the entries in the column `components$names` of the input object.

Last updates

2020-10-30, DM: Renamed from *simulate_OSLcurve* to *simulate_OSLcomponents*; Renamed argument from *template.curve* to *curve*; Rewrote roxygen documentation

Author(s)

Dirk Mittelstraß, <dirk.mittelstrass@luminescence.de>

Please cite the package the following way:

Mittelstraß, D., Schmidt, C., Beyer, J., Heitmann, J. and Straessner, A.: R package OSLdecomposition: Automated identification and separation of quartz CW-OSL signal components, *in preparation*.

References

Mittelstraß, D., 2019. Decomposition of weak optically stimulated luminescence signals and its application in retrospective dosimetry at quartz (Master thesis). TU Dresden, Dresden.

See Also

fit_OSLcurve, decompose_OSLcurve, plot_OSLcurve

Examples

```
# Set some arbitrary decay parameter for a dim CW-OSL measurement of quartz
components <- data.frame(name = c("fast", "medium", "slow"),
                         lambda = c(2, 0.5, 0.02),
                         n = c(1000, 1000, 10000))

# Simulate the CW-OSL curve and add some signal noise
curve <- simulate_OSLcomponents(components, simulate.curve = TRUE, add.poisson.noise = TRUE)

# Display the simulated curve
plot_OSLcurve(curve, components)
```

sum_OSLcurves *Combine RLum OSL records to one global average curve*

Description

This function adds up all CW-OSL records of the same type saved in Luminescence::RLum.Analysis objects and calculates the arithmetic mean signal from all records for each channel. This is useful to create global average curve with sufficient signal-to-noise ratio for OSL components identification with fit_OSLcurve or to create a signal background reference curve.

Usage

```
sum_OSLcurves(
  object,
  record_type = "OSL",
  aliquot_selection = NULL,
  offset_value = 0,
  verbose = TRUE,
  output.plot = FALSE,
  theme.set = ggplot2::theme_classic(),
  plot.first = FALSE,
  title = "default",
  filename = NULL
)
```

Arguments

`object`	Luminescence::RLum.Analysis or list of Luminescence::RLum.Analysis (**required**): Data set of one or multiple aliquots containing CW-OSL records.
`record_type`	character (*with default*): Type of records which are selected from the input `object`, for example: `"OSL"`,`"SGOSL"` or `"IRSL"`.
`aliquot_selection`	numeric vector (*optional*): Vector specifying the indices of elements (aliquots) of a list of Luminescence::RLum.Analysis objects which shall be included.
`offset_value`	numeric (*with default*): Signal offset (background) which will be subtracted from each record.
`verbose`	logical (*with default*): Enables console text output.
`output.plot`	logical (*with default*): returns a plot with all data points of all records and the average curve
`theme.set`	ggplot2::ggplot2-package object (*with default*): sets the graphical theme of the output plot. See ggplot2::theme_bw for available themes
`plot.first`	logical (*with default*): Plot includes additional drawing of first `record_type` record of first `object` list element.
`title`	character (*with default*): Plot title. Set `title = "default"` for an automatically generated title. Set `title = NULL` for no title.

filename — character (*optional*): File name or path to save the plot as image. If just a file name is given, the image is saved in the working directory. The image type is chosen by the file ending. Both, vector images as well as pixel images are possible. Allowed are `.pdf`, `.eps`, `.svg` (vector graphics), `.jpg`, `.png`, `.bmp` (pixel graphics) and more, see ggplot2::ggsave.

Value

A data.frame of the average CW-OSL curve is returned, containing two columns: `$time` and `$signal`.

Last updates

2020-10-30, DM: Overworked plotting; Expanded roxygen documentation

Author(s)

Dirk Mittelstraß, <dirk.mittelstrass@luminescence.de>

Please cite the package the following way:

Mittelstraß, D., Schmidt, C., Beyer, J., Heitmann, J. and Straessner, A.: R package OSLdecomposition: Automated identification and separation of quartz CW-OSL signal components, *in preparation*.

See Also

fit_OSLcurve, RLum.OSL_correction, RLum.OSL_global_fitting

Examples

```
# 'FB_10Gy' is a dose recovery test with the Fontainebleau quartz
# measured in a lexsyg research with green LED stimulation
data_path <- system.file("examples", "FB_10Gy_SAR.bin", package = "OSLdecomposition")
data_set <- Luminescence::read_BIN2R(data_path, fastForward = TRUE)

# Give average CW-OSL curve back
average_curve <- sum_OSLcurves(data_set)
```

Index

∗ **package**
OSLdecomposition-package, 2

character, *6*, *14, 15*, *17*, *20*, *23*, *26*, *30, 31*

data.frame, *4*, *6*, *9, 10*, *12*, *14*, *23*, *26–29*, *31*
decompose_OSLcurve, 3, *8*, *11–14*, *22*, *24*, *28, 29*
DEoptim::DEoptim, *8*, *11–13*, *27*
DEoptim::DEoptim.control, *10*, *12*

fit_OSLcurve, *5–7*, 7, *12, 13*, *15–18*, *25–31*

ggplot2::ggplot, *15*, *17*
ggplot2::ggplot2-package, *14, 15*, *17*, *30*
ggplot2::ggsave, *15*, *17*, *23*, *26*, *31*
ggplot2::theme_bw, *14*, *30*
ggplot2::theme_set, *14*
gridExtra::grid.arrange, *15*, *17*

knitr::kable, *26*

list, *10*, *16*, *20*, *23*, *26, 27*, *30*
logical, *6*, *9, 10*, *12*, *14, 15*, *17*, *20*, *23*, *26*, *28–30*
Luminescence::analyse_SAR.CWOSL, *22*, *24*
Luminescence::Luminescence-package, *18*, *22*, *25*
Luminescence::RLum.Analysis, *18*, *20*, *22, 23*, *25, 26*, *30*
Luminescence::RLum.Data.Curve, *6*, *9*, *12*, *14*, *24*

matrix, *6*, *9*, *12*, *14*
minpack.lm::nlsLM, *4, 5*, *7, 8*, *11*, *27*

nls, *10*
numeric, *5, 6*, *9*, *12*, *16, 17*, *20*, *23*, *26*, *28–30*

optimise_OSLintervals, *4*, *6, 7*, 11, *22*, *24*
OSLdecomposition (OSLdecomposition-package), 2
OSLdecomposition-package, 2

plot_OSLcurve, *11*, 13, *28, 29*
plot_PhotoCrosssections, *10, 11*, 16, *27*

RLum.OSL_correction, 18, *22*, *25*, *27*, *31*
RLum.OSL_decomposition, *7*, *11*, *13*, *21*, 22, *25*, *27*
RLum.OSL_global_fitting, *15*, *18*, *21, 22*, *24*, 25, *31*
rmarkdown::rmarkdown-package, *22, 23*, *25, 26*

simulate_OSLcomponents, *15*, 28
sum_OSLcurves, *6*, *9*, *11*, *19–22*, *25*, *27*, 30

vector, *10*