RLumModel - Getting started with RLumModel

Johannes Friedrich

2017-04-03

Contents

1 Introduction

This vignette shows a few examples for the **R**-package 'RLumModel'. The main function `model_LuminescenceSignals()` and their arguments will be explained. All calculations were done with 'RLumModel' 0.2.0 and 'Luminescence' 0.7.4.

2 Object structure of RLumModel

The output from the main function `model_LuminescenceSignals()` is of class `RLum.Analysis` (Kreutzer et al., 2012) and contains data of class `RLum.Data.Curve` in the slot 'records'. The advantage of this infrastructure is that the package 'Luminescence' offers a lot of methods to visualize and manipulate data.

All simualted data are stored in the slot 'records': TL/OSL/RF curves as well as the concentrations of every energy level from every step.

The following code loads a data set provided by the 'RLumModel' package and shows how to seperate TL/OSL/RF data from concentrations and how to visualize them.

```
data("ExampleData.ModelOutput", package = "RLumModel")

##show class
class(model.output)

##show structure
Luminescence::structure_RLum(model.output)
```

```
##seperate TL-curve from TL-concentrations
TL_curve <- Luminescence::get_RLum(model.output, recordType = "TL$")
TL_conc <- Luminescence::get_RLum(model.output, recordType = "(TL)", drop = FALSE)

##also possible: TL_curve <- get_RLum(model.output, record.id = 1)

##plot results
Luminescence::plot_RLum(TL_curve)
Luminescence::plot_RLum(TL_conc)
```

Some notes to the code example above:

- in 'TL_curve <- ...' appears "TL$". This is necessary to match the pattern "TL" without any sign after "TL", e.g. a bracket. The brackets are used (by default) for the concentrations.
- in 'TL_conc <- ...' the pattern "(TL)" will match all concentrations with "(TL)", see structure.
- `drop = FALSE` was used to keep the `RLum.Analysis` class for 'TL_conc'.
- To see a single plot of every energy-level, use the option `plot.single = TRUE` in `plot_RLum()`. For more details see the manual of 'Luminescence'.

```
##plot every energy-level by an extra plot
Luminescence::plot_RLum(TL_conc, plot.single = TRUE)
```

It is also possible to choose a `RLum.Data.Curve` by their 'record.id', which can be seen with:

```
##see structure of model.output
Luminescence::structure_RLum(model.output)
```

3 Selecting a quartz luminescence model

The first argument required for the function `model_LuminescenceSignals()` is the name of a quartz luminescence model to be used, respectively the used parameter set in this quartz luminescence model. All currently implemented quartz luminescence models were described in Friedrich et al. (2016). The command to select a set of parameters from a specific model in `RLumModel` is a character string with the name of the author and the year, e.g.

```
model <- "Bailey2001"
```

The available models are "Bailey2001", "Bailey2002", "Bailey2004", "Pagonis2007", "Pagonis2008" and "Friedrich2017" (Bailey (2001), Bailey (2002), Bailey (2004), Pagonis et al. (2007), V. Pagonis et al. (2008), Friedrich et al. (2017)).

The corresponding parameter set will be loaded automatically with the function call.

4 Creating a sequence

The second argument in the `model_LuminescenceSignals()` function is the sequence to be simulated. There are three different ways of creating a sequence.

For all sequences, temperature differences between sequence steps are automatically simulated with a heating or cooling step in between. Also, after irradiating the sample, it is automatically kept at irradiation temperature for further 5 s to allow the system to relax prior to the next step (Bailey, 2001).

4.1 Risø SEQ files

The first one is to use the popular and freely available Risø *Sequence Editor version 4.36*[1] to build a personal sequence and to save it as a SEQ-file (*.seq). Files created by the Sequence Editor can be imported directly using the path of the SEQ-file. The package comes along with an example SEQ-file in the package folder in 'extdata'. Thus, a potential sequence is created with

```
sequence <- system.file(
  "extdata",
  "example_SAR_cycle.SEQ",
  package = "RLumModel")
```

or wherever the SEQ-file is stored. While in the Sequence Editor irradiation is commonly defined in seconds, performing the simulation requires a dose transformation to gray. Therefore, the function `model_LuminescenceSignals()` offers a special argument called `lab.dose_rate`, representing the dose rate of the irradiation unit in the laboratory. By default, this dose rate is $1\,\mathrm{Gy\,s^{-1}}$, but can be modified, e.g.,

```
lab.dose_rate <-  0.105
```

4.2 Keywords

The second way of creating a sequence is by referring to a list with keywords and a certain order of code numbers or named values, which are shown in Table 1. With these keywords, it is possible to create quickly an R object of type list, which can be read by the `model_LuminescenceSignals()` function.

ARGUMENTS	DESCRIPTION	SUB-ARGUMENTS
`TL`	Thermally stimulated luminescence	'temp_begin' [°C], 'temp_end' [°C], 'heating_rate' [°C/s]
`OSL`	Optically stimulated luminescence	'temp' [°C], 'duration' [s], 'optical_power' [%]
`ILL`	Illumination	'temp' [°C], 'duration' [s], 'optical_power' [%]
`LM_OSL`	Linear modulated OSL	'temp' [°C], 'duration' [s], optional: 'start_power' [%], 'end_power' [%]
`RF`	Radiofluorescence	'temp' [°C], 'dose' [Gy], 'dose_rate' [Gy/s]
`IRR`	Irradiation	'temp' [°C], 'dose' [Gy], 'dose_rate' [Gy/s]
`RF_heating`	RF during heating/cooling	'temp_begin' [°C], 'temp_end' [°C], 'heating_rate' [°C/s], 'dose_rate' [Gy/s]
`CH`	Cutheat	'temp' [°C], optional: 'duration' [s], 'heating_rate' [°C/s]
`PH`	Preheat	'temp' [°C], 'duration' [s], optional: 'heating_rate' [°C/s]
`PAUSE`	Pause	'temp' [°C], 'duration [s]'

Table 1: Keywords for creating a sequence in 'RLumModel'. Note that 100 % optical power equates to 20 mW $\mathrm{cm^{-2}}$. Of course, values > 100 % are allowed.

Some examples to this kind of sequence creating:

```
sequence <- list(
 IRR = c(temp = 20, dose = 10, dose_rate = 1),
 TL = c(temp_begin = 20, temp_end = 400 , heating_rate = 5))
```

This sequences describes an irradiation simulation at 20 °C with a dose of 10 Gy and a dose rate of 1 Gy/s, which is followed by a TL simulation from 20 °C to 400 °C with a heating rate of 5 °C/s. Note that it is important that for each sequence keyword like 'IRR' or 'TL' either the vector has to be named or the correct order of arguments is used, see 'sub-arguments' in Table 1. Thus the above mentioned code is equivalent to the following one:

```
sequence <- list(
  IRR = c(20, 10, 1),
  TL = c(20, 400, 5))
```

[1] http://www.nutech.dtu.dk/english/Products-and-Services/Dosimetry/Radiation-Measurement-Instruments/TL_OSL_reader/Software; 2016-04-11

4.3 Creating a SAR/DRT sequence

However, to create a SAR or dose-recovery-test (DRT) sequence with the Risø Sequence Editor or with keywords is time-consuming, because it contains a lot of individual sequence steps (preheat, optical stimulation, irradiation, ...). Therefore, a third way was implemented in 'RLumModel' to create a (SAR) sequence after Murray and Wintle (2000) with the (required) keywords RegDose, TestDose, PH, CH and OSL temp. In addition to these keywords, the user is able to set more detailed parameters for the SAR sequence, see Table 2:

ABBREVIATION	DESCRIPTION	EXAMPLE ARGUMENTS
`RegDose`	Dose points of the regenerative cycles [Gy]	c(0, 80, 140, 260, 320, 0, 80)
`TestDose`	Test dose for the SAR cycles [Gy]	50
`PH`	Temperature of the preheat [°C]	240
`CH`	Temperature of the cutheat [°C]	200
`OSL_temp`	Temperature of OSL read out [°C]	125
`OSL_duration`	Duration of OSL read out [s]	default: 40
`Irr_temp`	Temperature of irradiation [°C]	default: 20
`PH_duration`	Duration of the preheat [s]	default: 10
`dose_rate`	Dose rate of the laboratory irradiation source [Gy s^{-1}]	default: 1
`optical_power`	Percentage of the full illumination power [%]	default: 90
`Irr_2recover`	Dose to be recovered in a dose-recovery-test [Gy]	20

Table 2: Keywords for creating a SAR sequence with 'RLumModel'. The keyword `Irr_2recover` is only necessary for creating a DRT sequence. Note that 100 % optical power equates to 20 mW cm^{-2}. Of course, values > 100 % are allowed.

So a possible DRT sequence could be the next code example:

```
sequence <- list(
  RegDose = c(0,10,20,50,90,0,10),
  TestDose = 2,
  PH = 220,
  CH = 220,
  OSL_temp = 125,
  Irr_2recover = 20)
```

This sequence describes a DRT, where a dose of 20 Gy will be recovered with this test. The regenerative doses are definded as 0 (natural), 10 Gy, 20 Gy, 50 Gy, 90 Gy and for recuperation and recycling ratio 0 Gy and 10 Gy, respectively. The test dose is definded as 2 Gy. Preheat and Cutheat are at 220 °C and all OSL measurements are simualted at 125 °C. There are more options to set, see Table 2.

The RLumModel function `model_LuminescenceSignals()` is able to interprete this (sequence-) list as a DRT sequence.

5 Working examples

5.1 Simulate a TL measurement

First of all, a sequence is needed, which produces a TL signal after the sample has received a dose:

```
sequence <- list (
IRR = c (20 , 10 , 1) ,
TL = c (20 , 400 , 5))
```

Here, at a temperature of 20 °C a dose of 10 Gy was applied with a dose rate of 1 Gy/s followed by a TL measurement from 20 °C to 400 °C with a heating rate of 5 °C/s. Running this sequence with the `model_LuminescenceSignals()` function produces a model output:

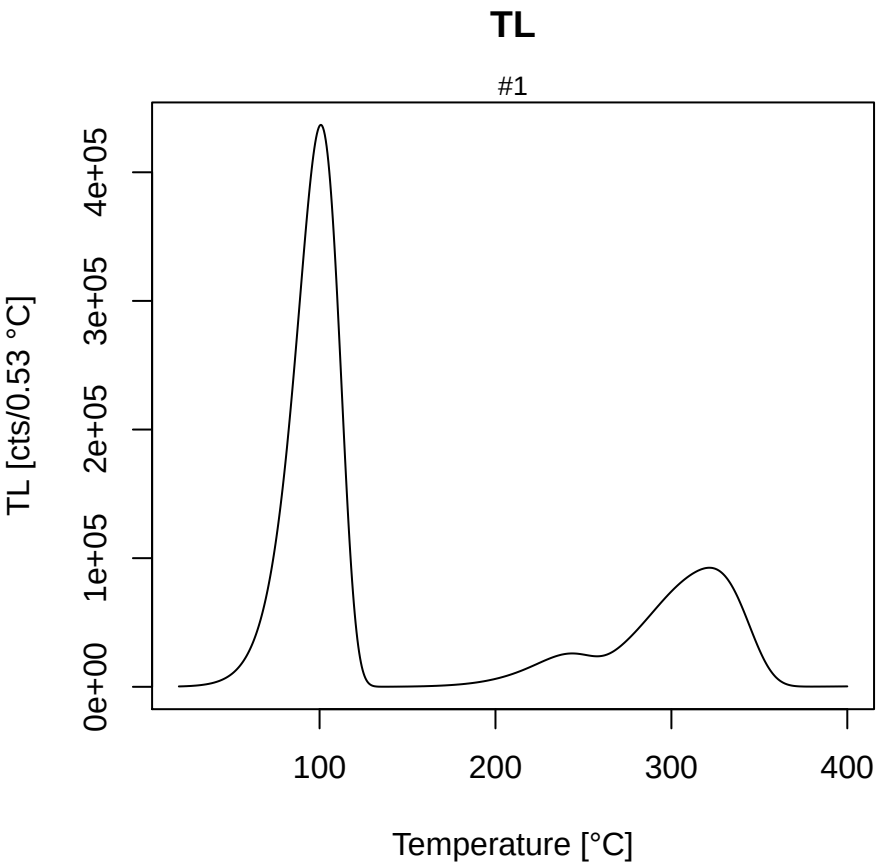

Figure 1: TL curve with parameter set 'Bailey2001' after 10 Gy laboratory dose

```
model.output <- model_LuminescenceSignals(
  model = "Bailey2001",
  sequence = sequence,
  verbose = FALSE)
```

This results in a TL curve like the one published in (Bailey (2001), Fig. 1), see figure above. In a further step, it is easy to produce known TL phenomena like the shift of the TL peak with varying heating rate. For this purpose, a loop over a TL simulation changes the heating rate in each run.

```
##set heating rate
heating.rate <- seq(from = 2, to = 10, by = 2)

##model signals
##"verbose = FALSE" for no terminal output
## "TL$" for exact matching TL and not (TL)
model.output <- lapply(heating.rate, function(x){
  sequence <- list(
   IRR = c(20, 10, 1),
   TL = c(20, 400, x))


  TL_data <- model_LuminescenceSignals(
      sequence = sequence,
      model = "Bailey2001",
```

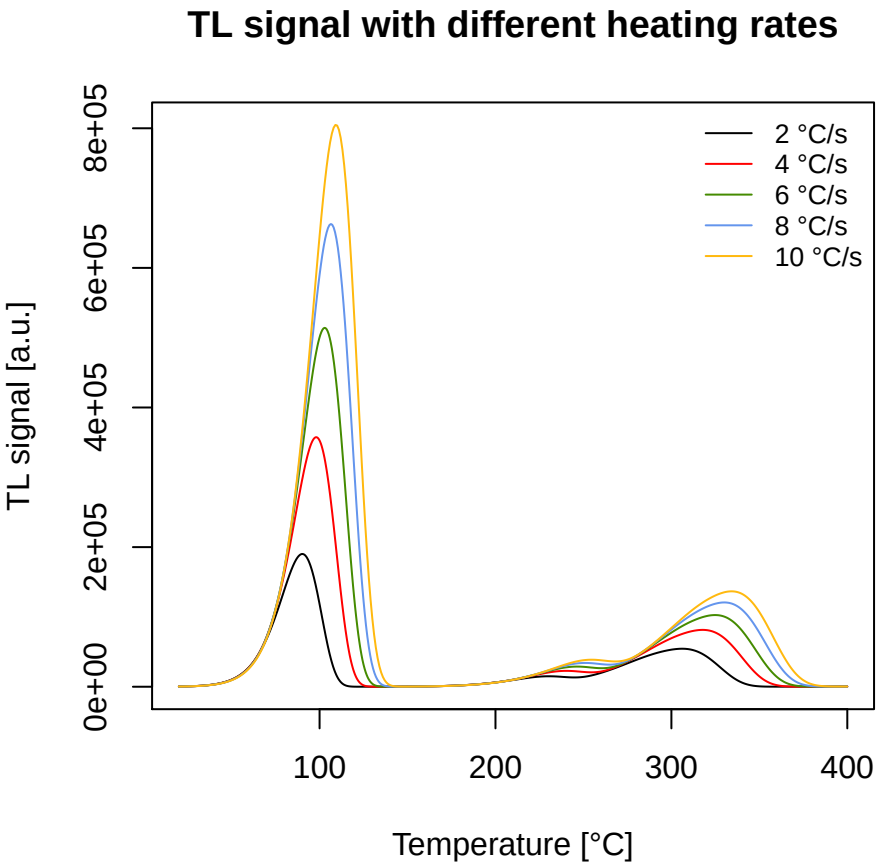

Figure 2: TL signal with different heating rates

```
      plot = FALSE,
      verbose = FALSE)

  return(Luminescence::get_RLum(TL_data, recordType = "TL$", drop = FALSE))

})

##merge output
model.output.merged <- merge_RLum(model.output)

##plot results
plot_RLum(
 object = model.output.merged,
 xlab = "Temperature [\u00B0C]",
 ylab = "TL signal [a.u.]",
 main = "TL signal with different heating rates",
 legend.text = paste(heating.rate, "°C/s"),
 combine = TRUE)
```

Some notes to the code above:

- the return of the lapply function is a `RLum.Analysis` object, because of drop = FALSE
- `recordType = TL$` is necessary to match the character TL exact and not the concentrations
- `merge_RLum` is necessary to merge all the single `RLum.Analysis` objects in the list 'model.output'

together to one `RLum.Analysis` object
- to see the results with another parameter set, only `model = "..."` has to be changed (see Sec. 2)

5.2 Simulating thermal activation characteristics (TACs)

Another frequently simulated phenomenon is the sensitisation of quartz TL by β- or γ-irradiation followed by activation at high temperatures (Zimmerman, 1971, Pagonis et al. (2003), V Pagonis et al. (2008), Adamiec et al. (2004)), termed as thermal activation characteristics (TACs). For a simulation sequence, the reader is referred to V Pagonis et al. (2008), Tab. 1. To simulate this phenomenon with the `model_LuminescenceSignals()` function, a loop causing a stepwise increase of the activation temperature is needed. The resulting intensity of the 110 °C TL peak can be plotted against the activation temperature, which shows TAC for the model parameters of "Pagonis2007".

```
##set temperature
act.temp <- seq(from = 80, to = 600, by = 20)

##loop over temperature
model.output <- vapply(X = act.temp, FUN = function(x) {

  ##set sequence, note: sequence includes sample history
  sequence <- list(
    IRR = c(20, 1, 1e-11),
    IRR = c(20, 10, 1),
    PH = c(x, 1),
    IRR = c(20, 0.1, 1),
    TL = c(20, 150, 5)
    )

  ##run simulation
  temp <- model_LuminescenceSignals(
    sequence = sequence,
    model = "Pagonis2007",
    simulate_sample_history = TRUE,
    plot = FALSE,
    verbose = FALSE
    )

  ## "TL$" for exact matching TL and not (TL)
  TL_curve <- Luminescence::get_RLum(temp, recordType = "TL$")

  ##return max value in TL curve
  return(max(get_RLum(TL_curve)[,2]))

}, FUN.VALUE = 1)
```

5.3 Simulating dependency of the OSL signal on the illumination power density

The function `model_LuminescenceSignals()` is also capable of simulating OSL phenomena. The next figure shows the dependency of the OSL signal on the power density of illumination for the model "Bailey2004".

```
##set optical power [%]
optical_power <- c(0,20,40,60,80,100)
```

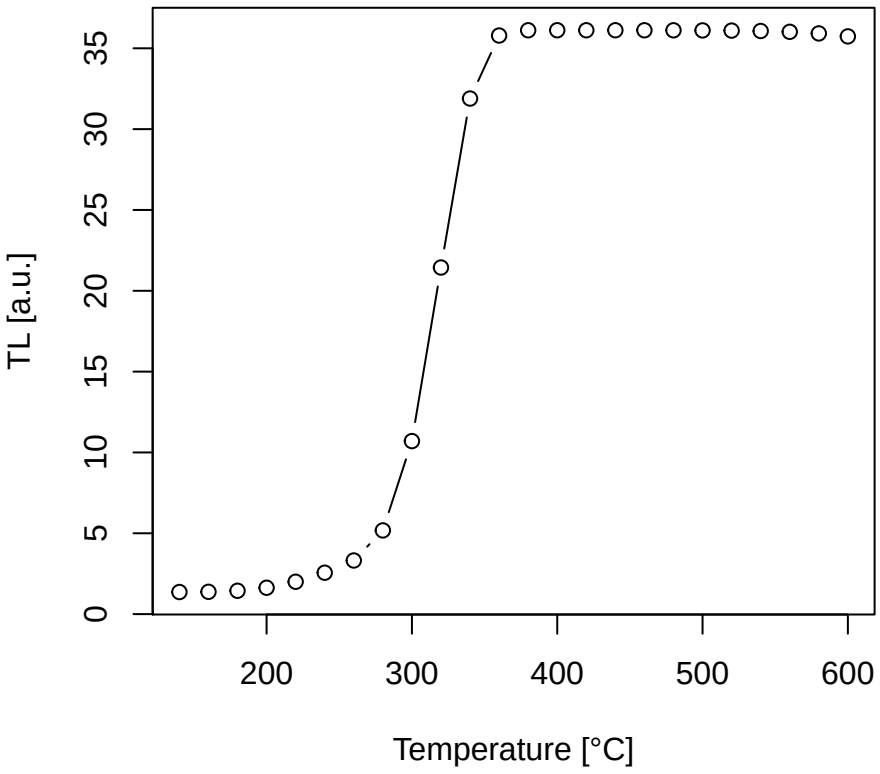

Figure 3: TAC with parameter set of 'Pagonis2007'

```
##loop over power
model.output <- lapply(optical_power, function(x){

  ##set sequence
  sequence <- list(
    IRR = c(20, 50, 1),
    PH = c(220, 10, 5),
    OSL = c(125, 50, x))


  data <-  model_LuminescenceSignals(
    sequence = sequence,
    model = "Bailey2004",
    plot = FALSE,
    verbose = FALSE)

  ##"OSL$" for exact matching OSL and not (OSL)
  return(Luminescence::get_RLum(data, recordType = "OSL$", drop = FALSE))

})

##merge output
model.output.merged <- Luminescence::merge_RLum(model.output)

##plot results
Luminescence::plot_RLum(
```

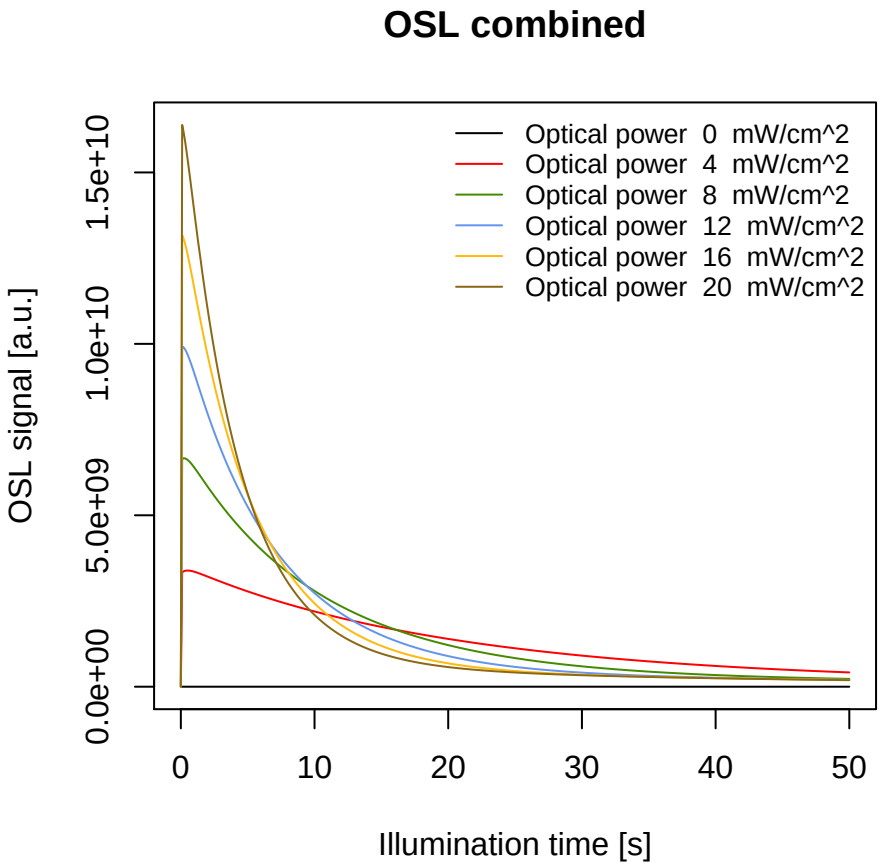

Figure 4: OSL measurement with different optical power densities with the parameter set of 'Bailey2004'

```
object = model.output.merged,
xlab = "Illumination time [s]",
ylab = "OSL signal [a.u.]",
legend.text = paste("Optical power ", 20 * optical_power / 100," mW/cm^2"),
combine = TRUE
)
```

5.4 Simulating and analysing SAR measurements

For simulating a DRT, it is necessary to define a sequence with the keyword `Irr_2recover`, as mentioned in Section 4.3.

It should be mentioned that a simulation of a combinded PHPT and DRT may be very time-consuming, because for every preheat temperature a complete SAR cycle has to be run. A typical DRT sequence featuring various PH temperatures in 'RLumModel' is listed below. Note that in such a DRT simulation a loop over different preheat temperatures has to be written, utilising characteristic parameters from the literature. The test dose is set to 10% and the regeneration dose points to 40%, 70%, 130%, 160%, 0% and 40% of the recovery dose.

The data created by 'RLumModel' can be directly passed to the functions `Luminescence::analyse_SAR.CWOSL()` and `Luminescence::plot_DRTResults()` for routine analyses and plotting.

```
##set PH temperatures
PH_temp <- seq(from = 160, to = 300, by = 20)

##set regeneration doses
RegDose = c(0, 80, 140, 260, 320, 0, 80)

##loop over PH temperatures
DRT.output <- lapply(PH_temp, function(x){

  sequence <- list(
       RegDose = RegDose,
       TestDose = 20,
       PH = x,
       CH = x,
       OSL_temp = 125,
       Irr_2recover = 200)

  model.output <- model_LuminescenceSignals(
       sequence = sequence,
       model = "Pagonis2008",
       plot = FALSE,
       verbose = FALSE)

  results <- Luminescence::analyse_SAR.CWOSL(
       object = model.output,
       signal.integral.min = 1,
       signal.integral.max = 7,
       background.integral.min = 301,
       background.integral.max = 401,
       fit.method = "EXP",
       dose.points = RegDose,
       plot = FALSE)

  temp <- get_RLum(results)
  out <- data.frame(
    De = temp$De,
    De.error = temp$De.Error)

  return(out)
})
```

```
## [plot_GrowthCurve()] Fit: EXP (interpolation) | De = 179.1 | D01 = 101.51
## [plot_GrowthCurve()] Fit: EXP (interpolation) | De = 179.46 | D01 = 101.46
## [plot_GrowthCurve()] Fit: EXP (interpolation) | De = 180.18 | D01 = 101.4
## [plot_GrowthCurve()] Fit: EXP (interpolation) | De = 180.6 | D01 = 101.41
## [plot_GrowthCurve()] Fit: EXP (interpolation) | De = 182.24 | D01 = 101.44
## [plot_GrowthCurve()] Fit: EXP (interpolation) | De = 179.85 | D01 = 102.26
## [plot_GrowthCurve()] Fit: EXP (interpolation) | De = 166.73 | D01 = 111.51
## [plot_GrowthCurve()] Fit: EXP (interpolation) | De = 159.15 | D01 = 161.2
```

```
##output as data.frame for plot_DRTResults
DRT.result <- as.data.frame(do.call(rbind, DRT.output))

##plot DRT.results
```

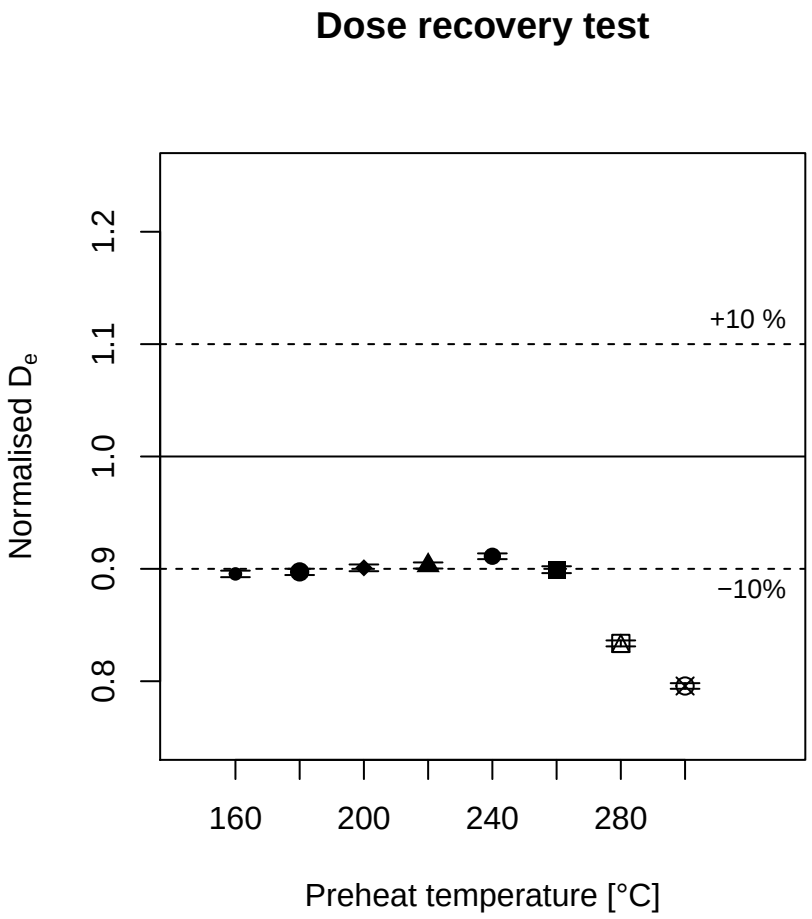

Figure 5: Dose recovery test (DRT) with the parameter set of 'Pagonis2008'

```
Luminescence::plot_DRTResults(
     DRT.result,
     preheat = PH_temp,
     given.dose = 200)
```

In the code above, `plot = FALSE` was chosen, because a single OSL plot is not necessary to analyse a SAR sequence. To calculate a D_e from the produced `RLum.Analysis` object 'model.output', the function `Luminescence::analyse_SAR.CWOSL()` is suitable. After specifying a number of evaluation parameters (signal and background integration interval, dose points and fit function for the dose response curve) and the analysis process, the reduced data are stored in an `RLum.Results` object, here termed 'results' . A background integration interval from 301 to 401 translates to the signal from 30 s to 40 s, because a channel has the default width of 0.1 s. Accordingly, the signal integral ranges from 0.1 s to 0.7 s.

```
##set RegDose
RegDose = c(0, 80, 140, 260, 320, 0, 80)

##set sequence
sequence <- list(
  RegDose = RegDose,
  TestDose = 20,
  PH = 220,
  CH = 220,
```

```
  OSL_temp = 125
  )

##model
model.output <- model_LuminescenceSignals(
  sequence = sequence,
  model = "Pagonis2008",
  plot = FALSE,
  verbose = FALSE
  )

##analyse SAR sequence and plot only the resulting growth curve
results <-Luminescence::analyse_SAR.CWOSL(
  model.output,
  signal.integral.min = 1,
  signal.integral.max = 7,
  background.integral.min = 301,
  background.integral.max = 401,
  fit.method = "EXP",
  dose.points = RegDose,
  verbose = FALSE,
  plot.single = c(6)
  )
```

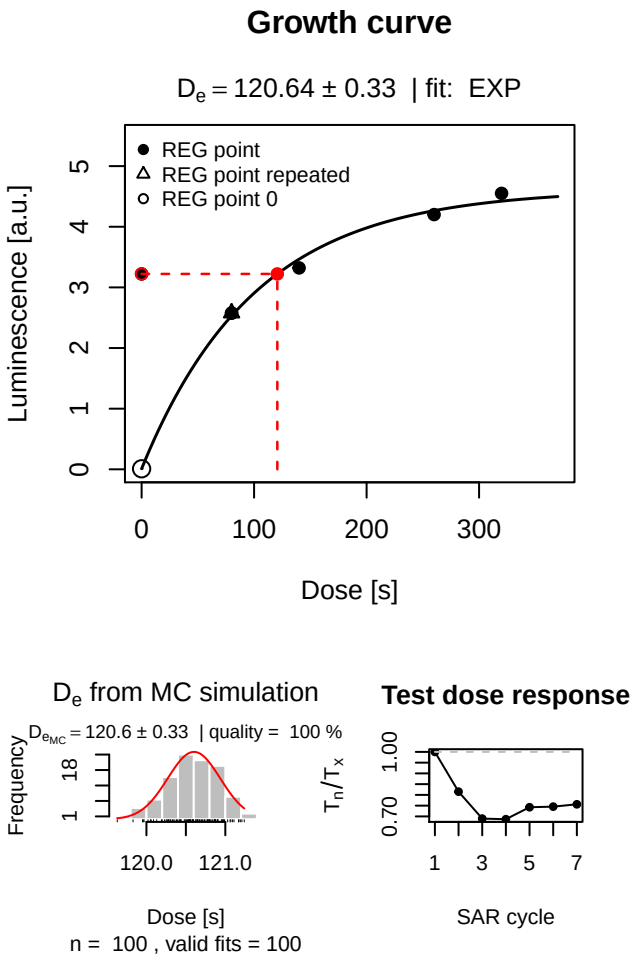

Figure 6: SAR protocol with the parameter set of 'Pagonis2008'

References

Adamiec, G., Garcia-Talavera, M., Bailey, R.M., La Torre, P.I. de, 2004. Application of a genetic algorithm to finding parameter values for numerical simulation of quartz luminescence. Geochronometria 23, 9–14.

Bailey, R., 2004. Paper I - Simulation of dose absorption in quartz over geological timescales and its implications for the precision and accuracy of optical dating. Radiation Measurements 38, 299–310.

Bailey, R., 2002. Simulations of variability in the luminescence characteristics of natural quartz and its implications for estimates of absorbed dose. Radiation Protection Dosimetry 100, 33–38.

Bailey, R.M., 2001. Towards a general kinetic model for optically and thermally stimulated luminescence of quartz. Radiation Measurements 33, 17–45.

Friedrich, J., Kreutzer, S., Schmidt, C., 2016. Solving ordinary differential equations to understand luminescence: 'RLumModel', an advanced research tool for simulating luminescence in quartz using R. Quaternary Geochronology 35, 88–100.

Friedrich, J., Pagonis, V., Chen, R., Kreutzer, S., Schmidt, C., 2017. Quartz radiofluorescence: A modelling approach. Journal of Luminescence 186, 318–325.

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, 1–8.

Pagonis, V., Balsamo, E., Barnold, C., Duling, K., McCole, S., 2008. Simulations of the predose technique for retrospective dosimetry and authenticity testing. Radiation Measurements 43, 1343–1353.

Pagonis, V., Chen, R., Wintle, A., 2007. Modelling thermal transfer in optically stimulated luminescence of quartz. Journal of Physics D: Applied Physics 40, 998.

Pagonis, V., Kitis, G., Chen, R., 2003. Applicability of the Zimmerman predose model in the thermoluminescence of predosed and annealed synthetic quartz samples. Radiation Measurements 37, 267–274.

Pagonis, V., Wintle, A., Chen, R., Wang, X., 2008. A theoretical model for a new dating protocol for quartz based on thermally transferred osl (tt-osl). Radiation Measurements 43, 704–708.

Zimmerman, J., 1971. The radiation-induced increase of the 100 ° C thermoluminescence sensitivity of fired quartz. Journal of Physics C: Solid State Physics 4, 3265–3276.