RLumModel - Getting started with RLumModel

Johannes Friedrich

2016-02-24

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.

2 Chosing a quartz luminescence model

In the R package "RLumModel" are five quartz luminescence parameter sets included. All these parameter sets are common in literature and often used to simulate luminescence phenomena.

The command to choose a set of parameters from a specific model in "RLumModel""" is simply a character string with the name of the author and the year, e.g.

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

The available models are "Bailey2001", "Bailey2002", "Bailey2004", "Pagonis2007" and "Pagonis2008". With this call the model parameter will automatically be loaded.

3 Creating a sequence

3.1 Import a Risø Sequence file

The first one is to use the popular and freely available Risø Sequence Editor version 4.36 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

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

3.2 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', 'temp_end', 'heating_rate'
OSL	Optically stimulated luminescence	'temp', 'duration', 'optical_power'
ILL	Illumination	'temp', 'duration', 'optical_power'
LM_OSL	Linear modulated OSL	'temp', 'duration', optional: 'start_power', 'end_power'
RF	Radiofluorescence	'temp', 'dose', 'dose_rate'
IRR	Irradiation	'temp', 'dose', 'dose_rate'
CH	Cutheat	'temp', optional: 'duration', 'heating_rate'
PH	Preheat	'temp', 'duration', optional: 'heating_rate'
PAUSE	Pause	'temp', 'duration'

Table 1: Keywords for creating a sequence in 'RLumModel'. Note that 100 % optical power equates to 20 mW 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))

sequence
```

```
## $IRR
##      temp      dose dose_rate
##        20        10         1
##
## $TL
##   temp_begin     temp_end heating_rate
##           20          400            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))
```

3.3 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, ir- radiation, ...). 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 key- words, 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.

4 Object structure of "model_LuminescenceSignals()"

The output from the main function "model_LuminescenceSignals()" is of class "RLum.Analysis" (see R-package Luminescence) 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
structure_RLum(model.output)

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

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

##plot results
plot_RLum(TL_curve)
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”, too.
- 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
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
structure_RLum(model.output)
```

5 Working examples

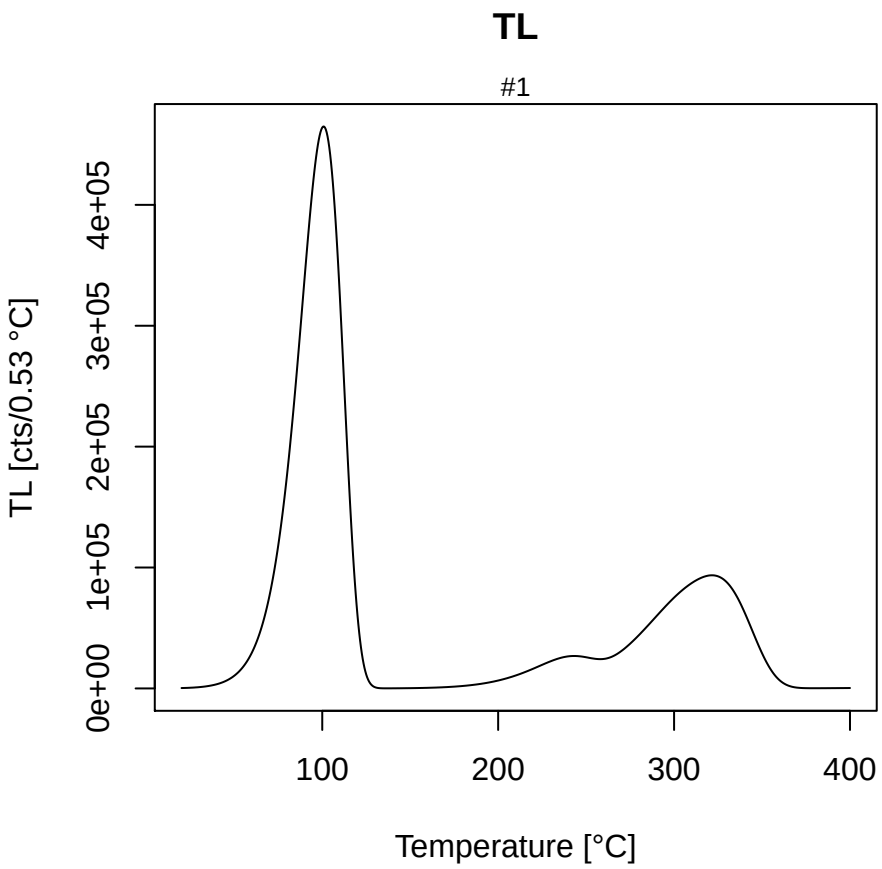
5.1 Simulate a TL measurement

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:

```
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(
 1:length(heating.rate), function(x){
  sequence <- list(
   IRR = c(20, 10, 1),
   TL = c(20, 400, heating.rate[x]))
```

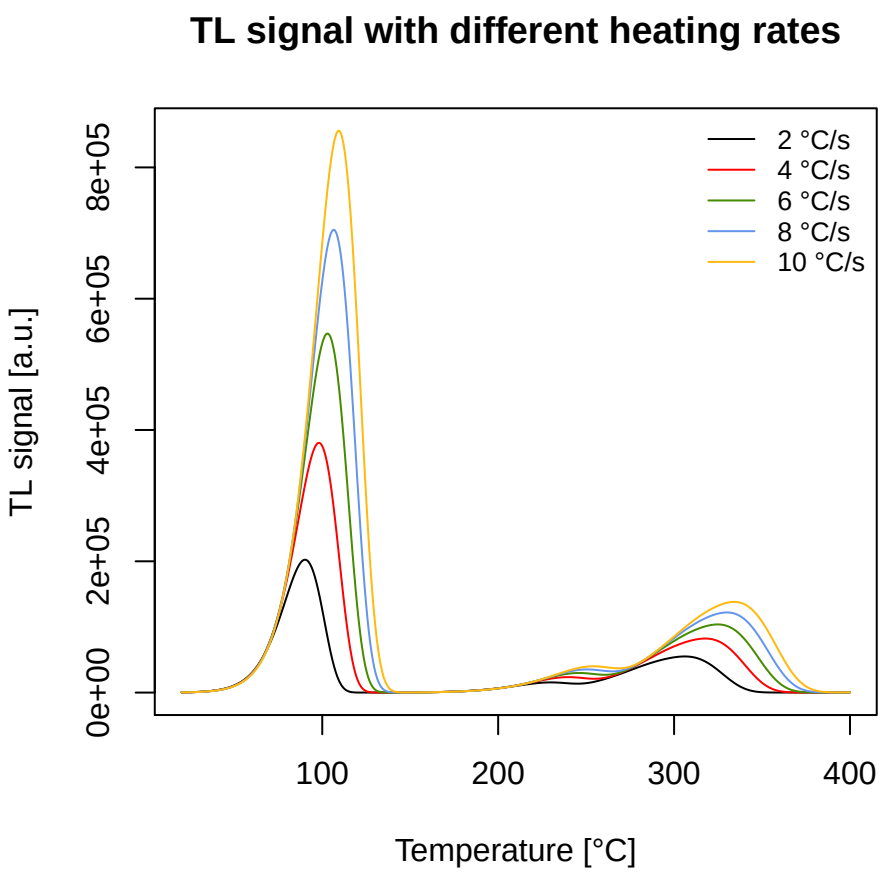
```
  TL_data <- model_LuminescenceSignals(
      sequence = sequence,
      model = "Bailey2001",
      plot = FALSE,
      verbose = FALSE)

  return(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)