RLumModel - Fitting model parameters to experimental data

Johannes Friedrich

2016-07-01

Contents

1 Introduction

This vignette gives an overview how to fit quartz luminescence model parameters to experimental data, TL and OSL, respectively.

The sensitivity analysis as well as the fitting procedure is done using the **R**-package 'FME' by Karline Soetaert and Thomas Petzoldt (https://cran.r-project.org/web/packages/FME/index.html). For further information the reader is referred to the FME manual and the FME vignettes therein.

2 Getting started

Before starting any analysis the data needed to be prepared to fit to the 'FME' and 'RLumModel' package. For creating a sequence or choosing a model the reader is referred to the RLumModel vignette 'RLumModel - Getting started with RLumModel'.

A new feature from RLumModel version $\geq$ 1.1 is the function `fit_RLumModel2data()`. The result from this function is the main function for all sensitivity analysis (local and global). The main arguments of the `fit_RLumModel2data()` function are a sequence to be simulated, a quartz luminescence model and the sequence step in 'sequence' to be analysed. Note that only sequence steps producing a signal output, are allowed. These are: TL, OSL, LM-OSL and RF.

3 Fitting TL data

First the global and local sensitivity of single parameters and then the procedure to fit model parameters to experimental TL data are shown.

3.1 Sensitivity analysis

The following example shows a TL simulation from 20 °C to 400 °C with a heating rate of 5 °C/s following an irradiation step at 20 °C, a dose of 5 Gy and a doserate of 0.5 Gy/s. The step to be analysed is the last TL step from 0 °C to 400°C with 5 °C/s and therefore the third step in the created sequence. As quartz luminescence model the parameter from Pagonis et al. (2007) are chosen. For sensitivity analysis the function `func_FME()` will be created with all mentioned arguments.

```
sequence <-  list(
    TL = c(20, 450, 5),
    IRR = c(20, 5, 0.5),
    TL = c(20, 450, 5))

model <- "Pagonis2007"

func_FME <- fit_RLumModel2data(
  sequence = sequence,
  model = model,
  seq.step2fit = 3)
```

3.1.1 Global Sensitivity Analysis

In global sensitivity analysis, certain parameters are changed over a large range, and the effect on certain model ouput variables assessed. Only a few preparations are needed to use of the FME function `sensRange()`:

- create parameters, which are able to be fitted with the function `sensRange()`. This is done via RLumModel function `extract_pars2FME()`. The user chooses only the used luminescence model.
- create a data.frame to specify the minimum and maximum values of parameters
- name the rowes of this data.frame like the used parameters

```
parms <- extract_pars2FME(model = model)


parRanges <- data.frame(min = c(5.1e7,  1e5, 1e9,  2.5e6,  5e8,   3e6,   1e8,  5e7,  1e6),
                        max = c(5.1e11, 1e9, 1e13, 2.5e10, 5e12,  3e10,  1e12, 5e11, 1e10))

rownames(parRanges) <- c("N1","N2","N3","N4","N5","N6","N7","N8","N9")
```

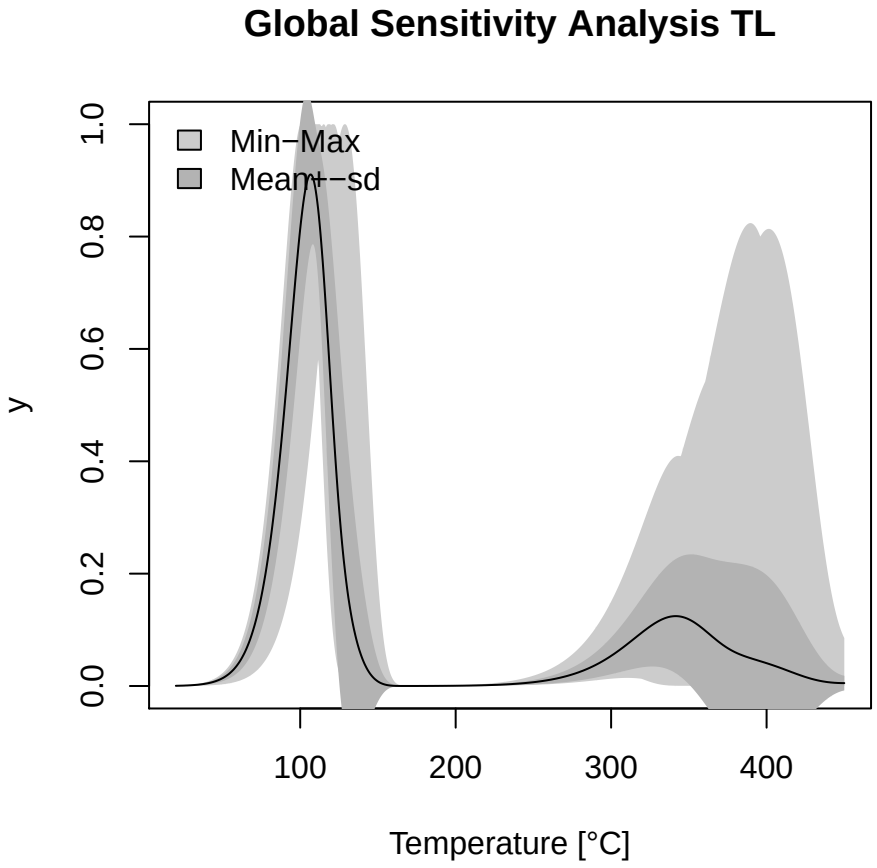
Now the user is able to calculate the global sensitivity of the chosen sequence-step and the parameters mentioned in `parRanges`. In FME this is done via function `sensRange()`. A short explanation of the parameters in sensRage in the example below:

- func: The above created function `func_FME()`
- parms: unlisted parameters (done via `extract_pars2FME()`)
- dist: The values of parRanges vary according to a regual grid

- The effect of the sensitivity 'signal' is desired. Note: According to inner structures the parameter sensvar always has to be 'signal'
- parRange is the minimum and maximum variation of the chosen parameters
- num: Times the model is run

For a detailed description the user is referred to the FME package. Subsequent a plot shows the range of possible TL curves.

```
global_Sens <- FME::sensRange(func = func_FME,
                    parms = parms,
                    dist = "latin",
                    sensvar = "signal",
                    parRange = parRanges,
                    num = 50)

global_Sens_sum <- summary(global_Sens)
```



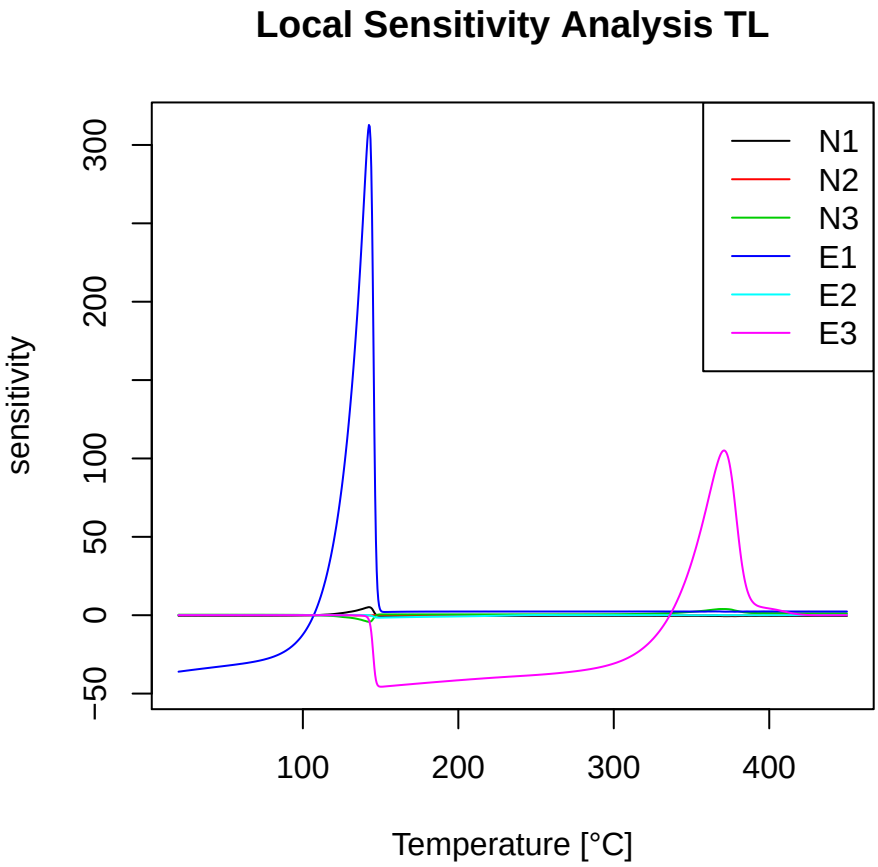
3.1.2 Local Sensitivity Analysis

In local sensitivity, the effect of a parameter value in a very small region near its nominal value is estimated.

With the above defined parms the user is now able run a local sensitivity analysis with the FME function `sensFun()`. In parameter 'senspar' the parameters to be analysed are written. In the example below parameters of N and E are observed. It is also possible to observe all parameters, then the argument 'senspar' is omitted.

```
SensR <- FME::sensFun(
    func = func_FME,
    parms = parms,
    senspar = c("N1", "N2", "N3", "E1", "E2", "E3"),
    varscale = NULL)

summary(SensR)
```

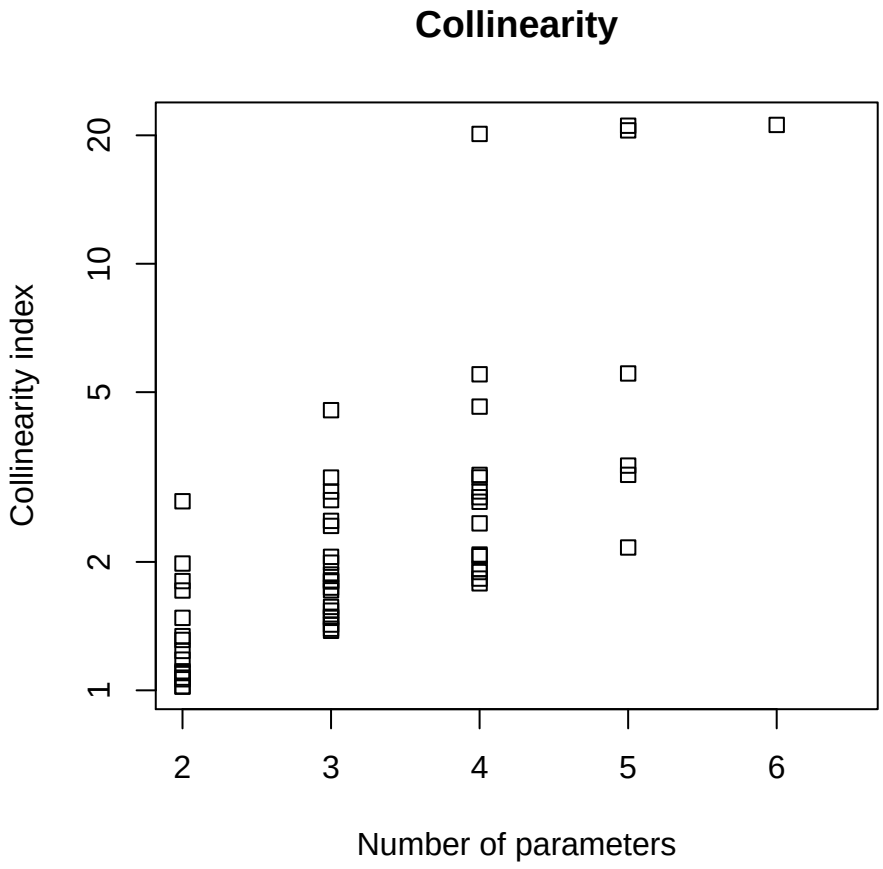
```
##       value   scale    L1     L2    Mean    Min    Max   N
## N1 5.1e+09 5.1e+09  0.57 0.0294 -0.2100  -0.61   5.14 861
## N2 1.0e+07 1.0e+07  0.02 0.0012  0.0023  -0.19   0.12 861
## N3 1.0e+11 1.0e+11  0.95 0.0442  0.6715  -4.21   3.95 861
## E1 9.7e-01 9.7e-01 18.38 1.6597  7.1883 -36.04 312.77 861
## E2 1.6e+00 1.6e+00  0.21 0.0154 -0.1699  -1.58   0.16 861
## E3 1.7e+00 1.7e+00 22.55 1.1181 -9.3865 -45.62 105.10 861
```



Based on the sensitivity functions of model variables to selection of parameters, a function called `collin()` calculates the collinearity or identifiability of sets of parameters.

The larger the collinearity value, the less identifiable the parameter based on the data. In general a collinearity value less than about 20 is 'identifiable'. Below we plot the collinarity as a function of the number of parameters selected. In this case all parameter sets are below this 'magic number'.

```
Coll <- FME::collin(SensR)

plot(Coll, log = "y")
```



3.2 Fit model parameters to experimental TL data

Now the fit from model parameters to experimental data is shown. A data set from **R**-package 'RlumModel' is loaded with a set of SAR-TL curves. The second TL curve (see subsection 3.1) is choosen and normalized. This normalization is recommended because it is difficult for the luminescence models to simulate absolute values but normalized signals are much easier to simualte. From function `func_FME()` automatically normalized data output is provided. It is important, that the name of the columns are 'time' and 'signal'. Otherwise the function call will break.

```
data("ExampleData.FittingTL", envir = environment())
exp_data <- get_RLum(TL_fitting_data, record.id = 2)
exp_data <- get_RLum(exp_data)

## normalize data
exp_data[,2] <- exp_data[,2]/max(exp_data[,2])
## rename columns
colnames(exp_data) <- c("time","signal")
```

The **R**-package FME offers the function `modCost()`, which estimates the 'model cost', which is the sum of (weighted) squared residuals of the model versus the data. This function is central to parameter identifiability analysis, model fitting or running a Markov chain Monte Carlo.

We first define an objective function `Fit_func()` that returns the residuals of the model versus the data, as estimated by modcost. It includes several fitting procedures; the default one is the Levenberg-Marquardt algorithm. Input to the function are the current values of the parameters that need to be finetuned and their

names. In this case all parameters $\neq 0$ are chosen to fit the experimental data. Note that this is not allways possible, because the collinearitiy has to be less than about 20.

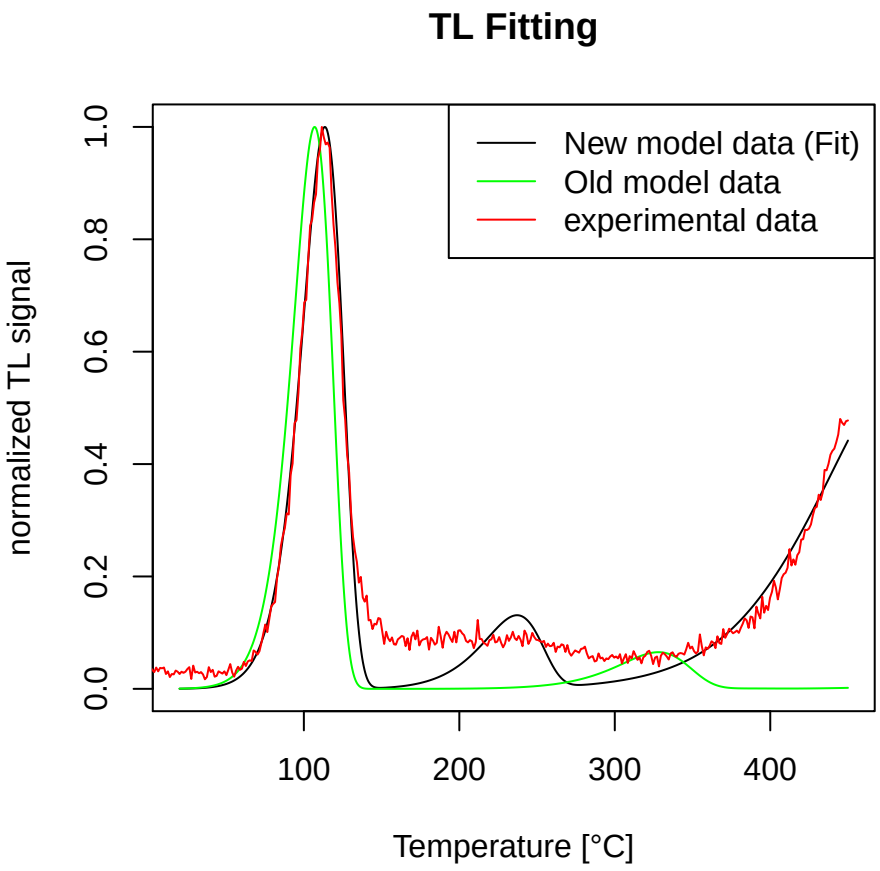
```
Fit_func <- function(x, parset = names(x)){

  parms[parset] <- exp(x)
  out <- func_FME(parms)
  return(modCost(model = out, obs = exp_data))
}

Fit_TL <- modFit(f = Fit_func,
                 p = log(parms[c("N1", "N2", "N3", "N4", "N5", "E1", "E2", "E3", "E4", "E5", "s1", "s2", "
                 method = "Port")
```

Comparing old and new model parameters show only small differences between the single model parameters.

```
##        old          new   percent
## N1 5.1e+09 6.775801e+09 132.85885
## N2 1.0e+07 9.948256e+06  99.48256
## N3 1.0e+11 3.973640e+10  39.73640
## N4 2.5e+08 2.484090e+08  99.36358
## N5 5.0e+10 1.117890e+11 223.57808
## E1 9.7e-01 9.734975e-01 100.36057
```

3.3 Plot results

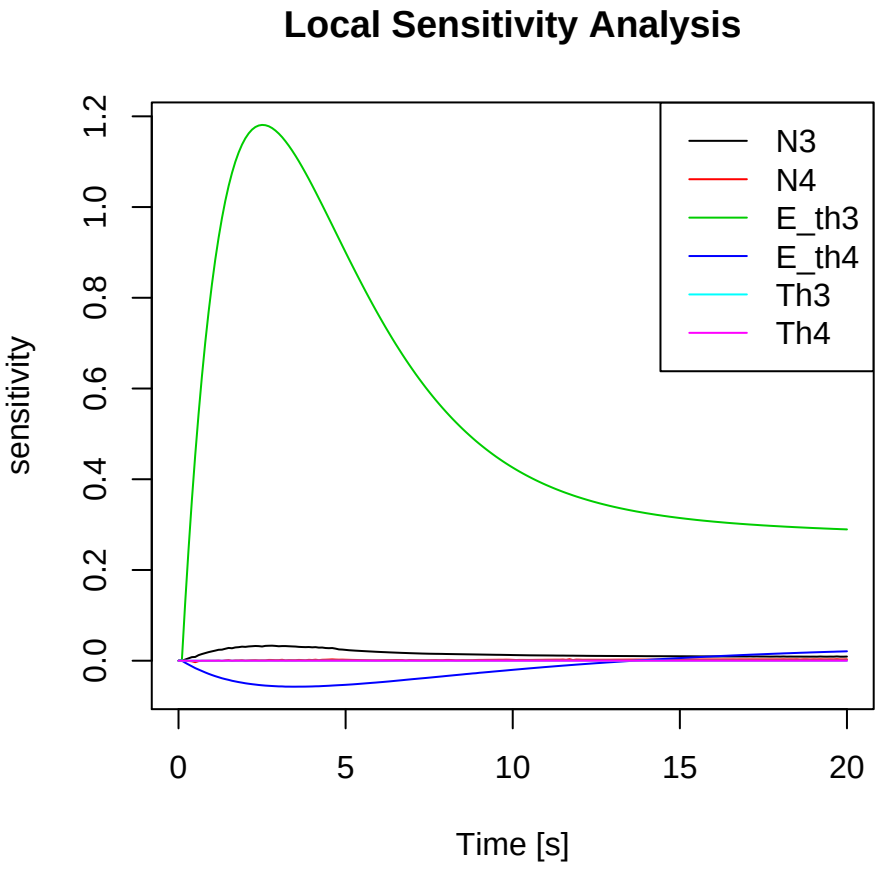
At least the simulations with old and new parameters and the experimental data are plotted.



4 Fitting OSL data

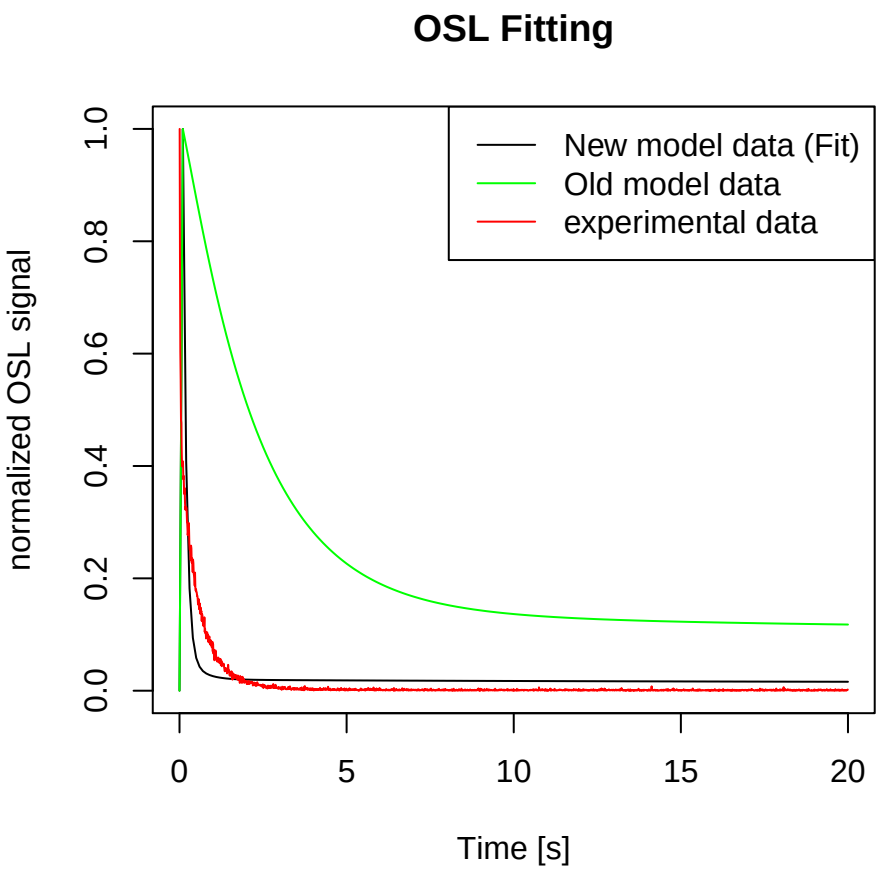
```
sequence <- list(OSL = c(125, 20, 200))

model <- "Bailey2004"

func_FME <- fit_RLumModel2data(
  sequence = sequence,
  model = model,
  seq.step2fit = 1)

parms <- extract_pars2FME(model = model)
```

```
SensR <- FME::sensFun(
  func = func_FME,
  parms = parms,
  sensvar = c("signal"),
  senspar = c("N3", "N4", "E_th3", "E_th4", "Th3", "Th4"),
  varscale = 1)
```




```
data(ExampleData.CW_OSL_Curve, envir = environment())
exp_data <- CW_Curve.BosWallinga2012

exp_data[,2] <- exp_data[,2]/max(exp_data[,2])
colnames(exp_data) <- c("time","signal")

Fit_func_OSL <- function(x, parset = names(x)){

  parms[parset] <- exp(x)
  out <- func_FME(parms)
  return(modCost(model = out, obs = exp_data))
}

Fit_OSL <- modFit(f = Fit_func,
              p = log(parms[c("N3", "N4", "E_th3", "Th3")]),
              method = "Port")
```

```
##            old          new      percent
## N3    2.05e+11 4.879490e+14 2.380239e+05
## N4    7.04e+10 3.808569e-02 5.409899e-11
## E_th3 1.00e-01 7.172034e-02 7.172034e+01
## Th3   1.00e-16 1.716299e-13 1.716299e+05
```



A Monte-Carlo methode

5 Summary

References

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