Getting started with RLumCarlo

Sebastian Kreutzer, Johannes Friedrich, Vasilis Pagonis, Christoph Schmidt

Last modified: 2019-10-08

Scope

RLumCarlo is collection of energy-band models to simulate luminescence signals using Monte-Carlo (MC) methods. This document aims at providing an overview and a brief introduction to RLumCarlo.

The models in RLumCarlo

The following tables lists the models implemented in RLumCarlo along with the R function call and the corresponding R and C++ files. The modelling takes place in the C++ functions which are wrapped by the R functions with a similar name. If you, however, want to cross-check the code, you should inspect files with the ending '.cpp'.

MODEL.NAME	R.CALL	FILES
MC_CW_IRSL_DELOC	run_MC_CW_IRSL_DELOC()	R/run_MC_CW_IRSL_DELOC.R src/MC_C_MC_CW_IRSL_DELOC.cpp
MC_CW_IRSL_LOC	run_MC_CW_IRSL_LOC()	R/run_MC_CW_IRSL_LOC.R src/MC_C_MC_CW_IRSL_LOC.cpp
MC_CW_IRSL_TUN	run_MC_CW_IRSL_TUN()	R/run_MC_CW_IRSL_TUN.R src/MC_C_MC_CW_IRSL_TUN.cpp
MC_ISO_DELOC	run_MC_ISO_DELOC()	R/run_MC_ISO_DELOC.R src/MC_C_MC_ISO_DELOC.cpp
MC_ISO_LOC	run_MC_ISO_LOC()	R/run_MC_ISO_LOC.R src/MC_C_MC_ISO_LOC.cpp
MC_ISO_TUN	run_MC_ISO_TUN()	R/run_MC_ISO_TUN.R src/MC_C_MC_ISO_TUN.cpp
MC_LM_OSL_DELOC	run_MC_LM_OSL_DELOC()	R/run_MC_LM_OSL_DELOC.R src/MC_C_MC_LM_OSL_DELOC.cpp
MC_LM_OSL_LOC	run_MC_LM_OSL_LOC()	R/run_MC_LM_OSL_LOC.R src/MC_C_MC_LM_OSL_LOC.cpp
MC_LM_OSL_TUN	run_MC_LM_OSL_TUN()	R/run_MC_LM_OSL_TUN.R src/MC_C_MC_LM_OSL_TUN.cpp
MC_TL_DELOC	run_MC_TL_DELOC()	R/run_MC_TL_DELOC.R src/MC_C_MC_TL_DELOC.cpp
MC_TL_LOC	run_MC_TL_LOC()	R/run_MC_TL_LOC.R src/MC_C_MC_TL_LOC.cpp
MC_TL_TUN	run_MC_TL_TUN()	R/run_MC_TL_TUN.R src/MC_C_MC_TL_TUN.cpp

Each model can be run by calling one of the **R** functions starting with `run_`. Currently three different model

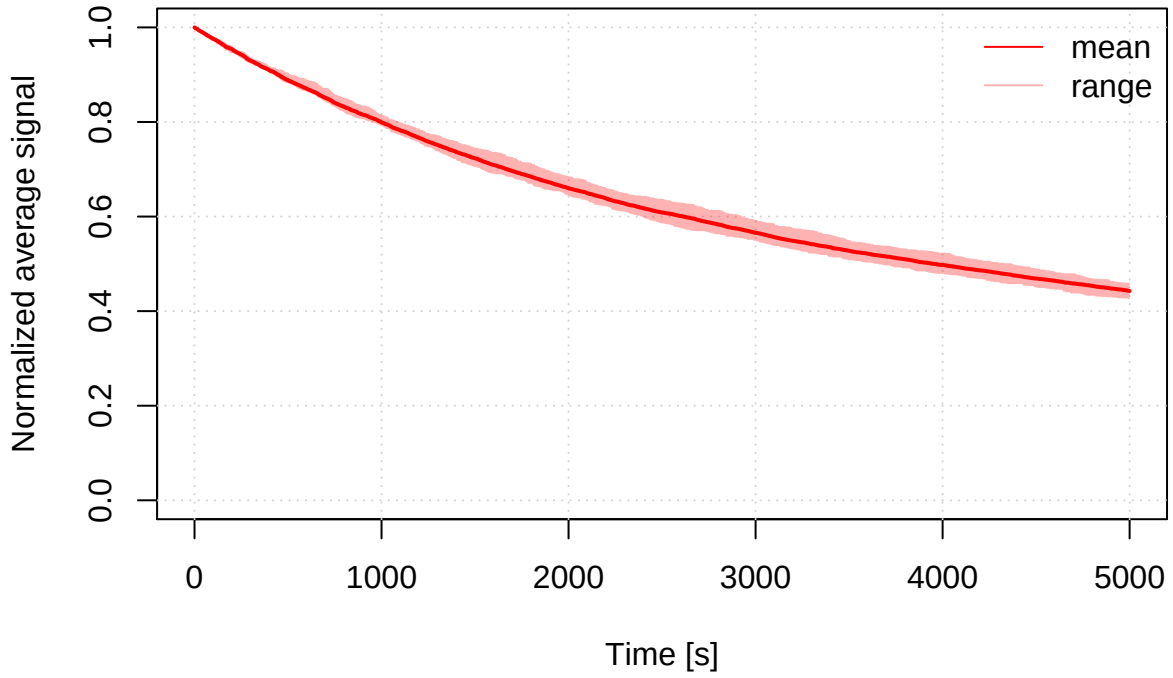
types (TUN: tunneling, LOC: localised transition, DELOC: delocalised transition) are implemented for the stimulation types TL, IRSL, LM-OSL, and ISO (isothermal). Please note that each model has different parameters and requirements.

Examples

Example 1: A first example

The first examples simulates an iso-thermal curve using the tunneling model.

```
results <- run_MC_ISO_TUN(
  E = 1.2,
  s = 1e10,
  T = 200,
  rho = 0.007,
  times = seq(0, 5000)
) %T>%
  plot_RLumCarlo(norm = TRUE, legend = TRUE)
```



The modelling output is an object of class `RLumCarlo_Model_Output`, which is basically a list consisting of an array and a vector.

```
str(results)
```

```
## List of 2
##  $ signal: num [1:5001, 1:21, 1:10] 0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "dimnames")=List of 3
##   .. ..$ : NULL
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ time  : int [1:5001] 0 1 2 3 4 5 6 7 8 9 ...
##  - attr(*, "class")= chr "RLumCarlo_Model_Output"
##  - attr(*, "model")= chr "run_MC_ISO_TUN"
```

While this represents the full modelling output results, the interpretation might be less straight forward and the user may want to condense the information via `summary()`. The function `summary()` is also used internally by the function `plot_RLumCarlo()`.

```
df <- summary(results)
```

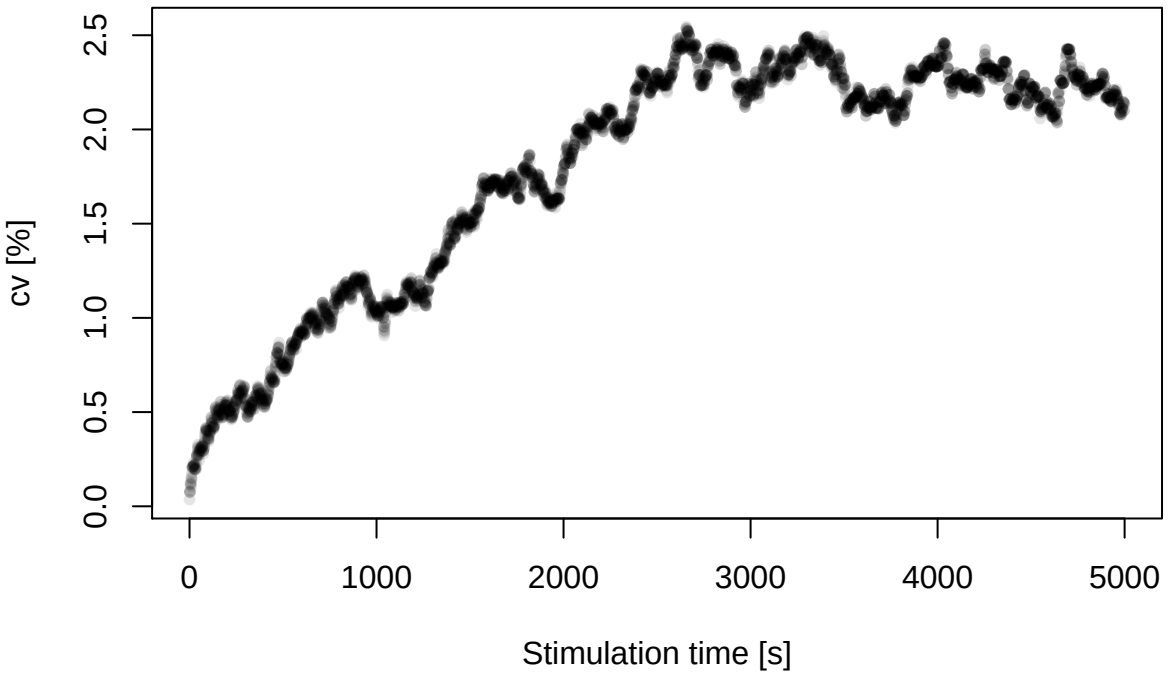
```
##       time           mean             y_min             y_max
##  Min.   :   0   Min.   :0.04671   Min.   :0.04484   Min.   :0.04847
##  1st Qu.:1250   1st Qu.:0.05409   1st Qu.:0.05195   1st Qu.:0.05651
##  Median :2500   Median :0.06421   Median :0.06174   Median :0.06730
##  Mean   :2500   Mean   :0.06812   Mean   :0.06638   Mean   :0.07044
##  3rd Qu.:3750   3rd Qu.:0.08005   3rd Qu.:0.07872   3rd Qu.:0.08199
##  Max.   :5000   Max.   :0.10554   Max.   :0.10547   Max.   :0.10557
##        sd                 var
##  Min.   :3.764e-05   Min.   :1.417e-09
##  1st Qu.:1.002e-03   1st Qu.:1.003e-06
##  Median :1.167e-03   Median :1.361e-06
##  Mean   :1.120e-03   Mean   :1.331e-06
##  3rd Qu.:1.327e-03   3rd Qu.:1.760e-06
##  Max.   :1.602e-03   Max.   :2.567e-06
```

```
head(df)
```

```
##   time      mean     y_min     y_max           sd          var
## 1    0 0.1055450 0.1054750 0.1055660 3.763821e-05 1.416635e-09
## 2    1 0.1055209 0.1053236 0.1055660 7.840228e-05 6.146917e-09
## 3    2 0.1054894 0.1053236 0.1055660 8.256933e-05 6.817694e-09
## 4    3 0.1054635 0.1053236 0.1055660 7.801698e-05 6.086649e-09
## 5    4 0.1054556 0.1053236 0.1055660 8.705057e-05 7.577802e-09
## 6    5 0.1054042 0.1052167 0.1055369 1.211621e-04 1.468026e-08
```

The call summarises the modelling results and returns a terminal output and a `data.frame` with, e.g., the mean or the standard deviation, which can be used to create plots for further insight. For instance, the stimulation time agains the relative standard deviation:

```
plot(
  x = df$time,
  y = (df$sd / df$mean) * 100,
  pch = 20,
  col = rgb(0,0,0,.1),
  xlab = "Stimulation time [s]",
  ylab = "cv [%]"
)
```



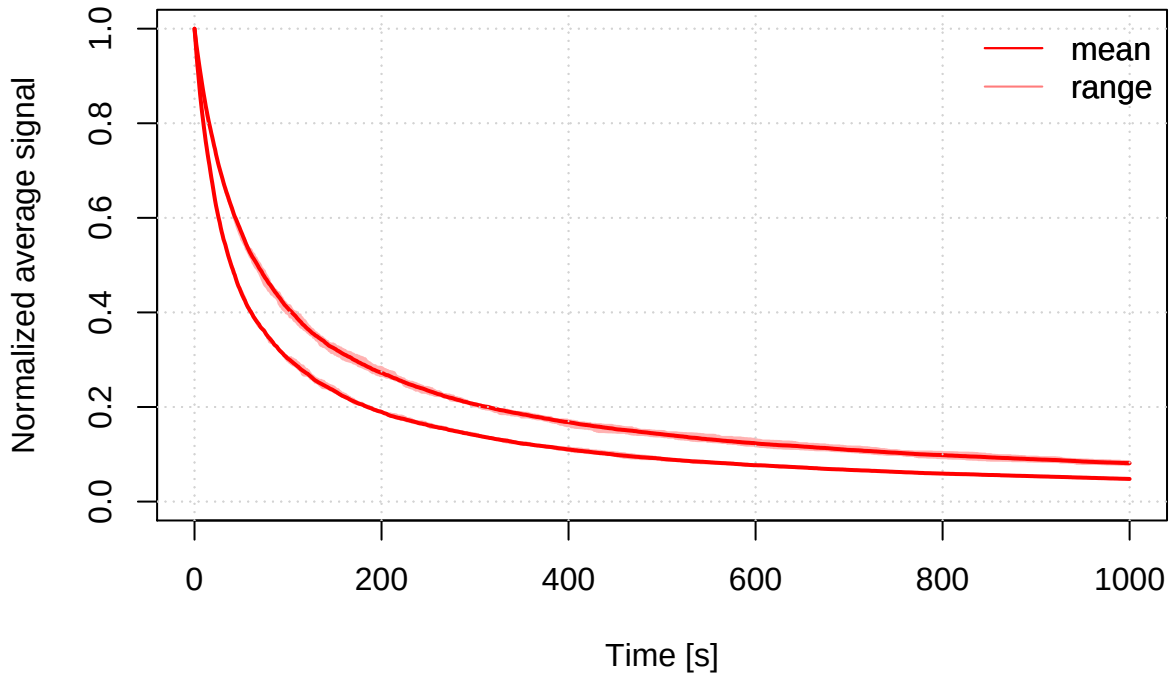
Example 2: Combining two plots

The following example uses continuous wave (CW) infrared light stimulation (IRSL), and combines two plots in one single plot window.

```
times <- seq(0, 1000)

## Run MC simulation
run_MC_CW_IRSL_TUN(A = 0.12, rho = 0.003, times = times) %>%
  plot_RLumCarlo(norm = TRUE, legend = TRUE)

run_MC_CW_IRSL_TUN(A = 0.21, rho = 0.003, times = times) %>%
  plot_RLumCarlo(norm = TRUE, add = TRUE)
```



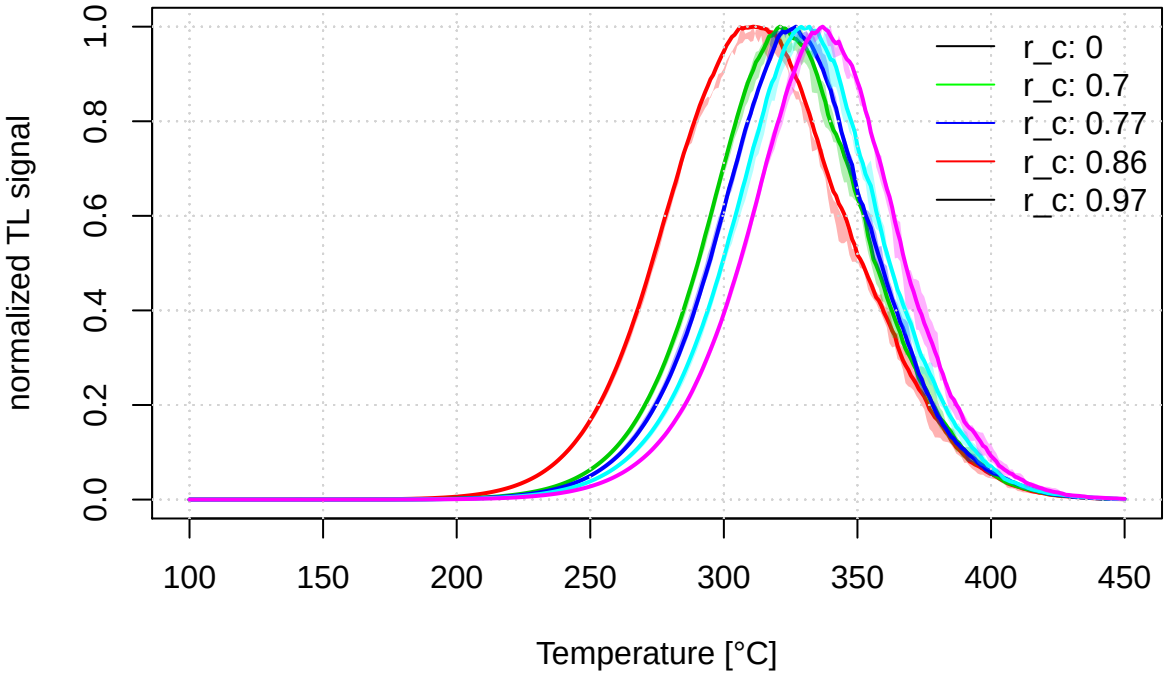
Example 3: Testing parameters

The example above can be further extended to test the effect of different parameters. Contrary to the example above, here the results are stored in a `list` and `plot_RLumCarlo()` is called only one time.

```
s <- 3.5e12
rho <- 0.015
E <- 1.45
r_c <- c(0,0.7,0.77,0.86, 0.97)
times <- seq(100, 450) # time = temperature
results <- lapply(r_c, function(x) {
  run_MC_TL_TUN(
    s = s,
    E = E,
    rho = rho,
    r_c = x,
    times = times
  )

})
```

```
## NULL
```


normalized TL signal
0.0
0.2
0.4
0.6
0.8
1.0
100
150
200
250
300
350
400
450
Temperature [°C]
r_c: 0
r_c: 0.7
r_c: 0.77
r_c: 0.86
r_c: 0.97