Getting started with RLumCarlo

Sebastian Kreutzer, Johannes Friedrich, Vasilis Pagonis, Christoph Schmidt

Last modified: 2019-10-09

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 (*.R) and C++ (*.cpp) 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_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_CW_OSL_DELOC	run_MC_CW_OSL_DELOC()	R/run_MC_CW_OSL_DELOC.R src/MC_C_MC_CW_OSL_DELOC.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.

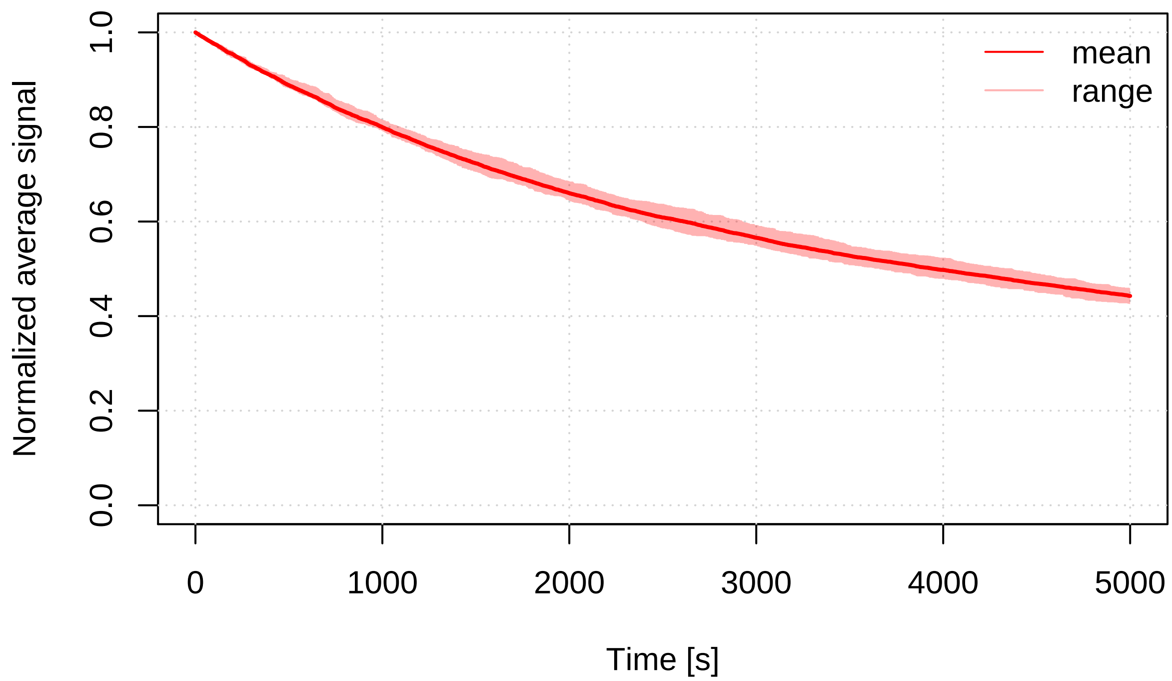
Examples

Example 1: A first example

The first examples simulates an iso-thermal curve using the tunneling model. Returned are either the simulated signal or the estimated remaining charges. The Function `plot_RLumCarlo()` provides an easy way to visualise the modelling results.

Modell the signal

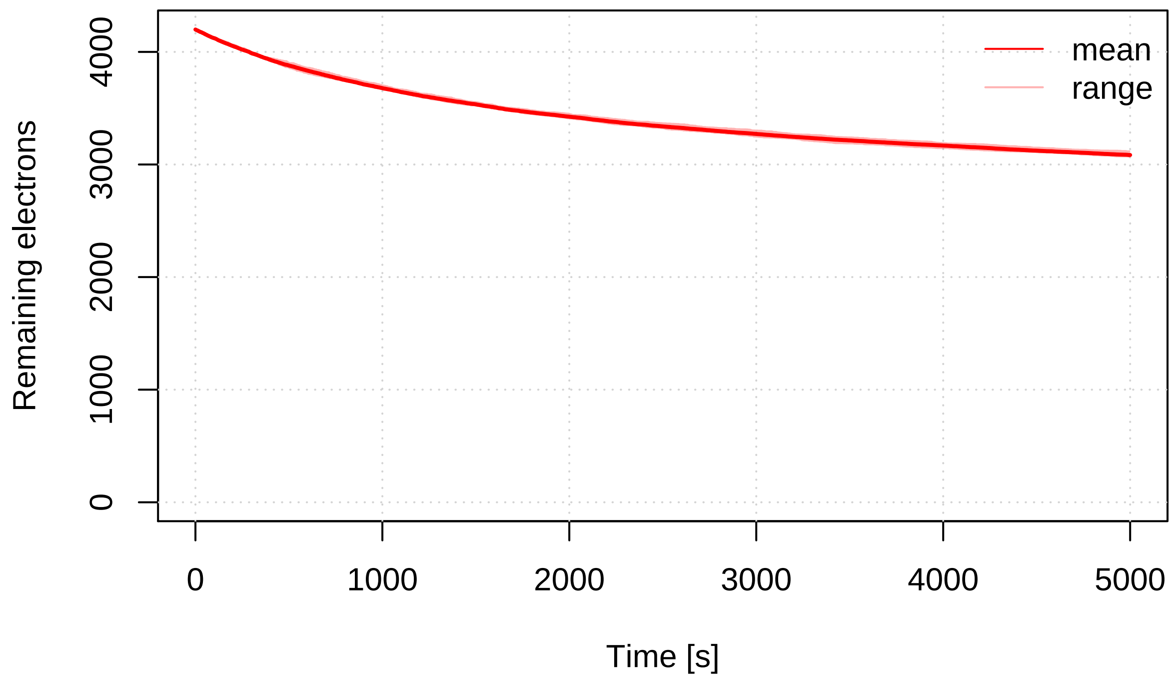
```
results <- run_MC_ISO_TUN(
  E = 1.2,
  s = 1e10,
  T = 200,
  rho = 0.007,
  times = seq(0, 5000)
) %T>%
  plot_RLumCarlo(norm = TRUE, legend = TRUE)
```



Modell remaining charges

```
results <- run_MC_ISO_TUN(
  E = 1.2,
  s = 1e10,
  T = 200,
  rho = 0.007,
  times = seq(0, 5000),
```

```r
  output = "remaining_e"
) %T>%
  plot_RLumCarlo(
    legend = TRUE,
    ylab = "Remaining electrons"
    )
```



Understanding the numerical output

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

```r
str(results)
```

```
## List of 2
##  $ signal: num [1:5001, 1:21, 1:10] 200 200 200 200 200 197 196 196 196 196 ...
##   ..- 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, its 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()`.

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

```
##       time           mean          y_min          y_max     
##  Min.   :   0   Min.   :3084   Min.   :3060   Min.   :3125  
##  1st Qu.:1250   1st Qu.:3190   1st Qu.:3159   1st Qu.:3224  
##  Median :2500   Median :3338   Median :3314   Median :3375  
```

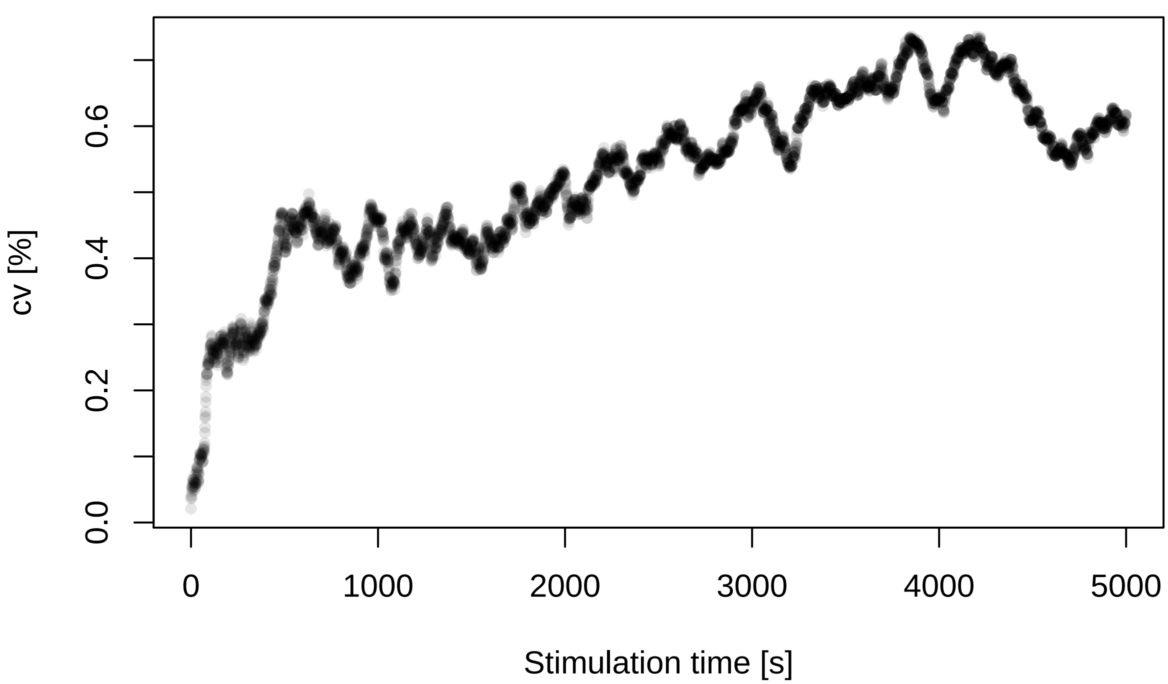
```
## Mean   :2500   Mean   :3424   Mean   :3399   Mean   :3457
## 3rd Qu.:3750   3rd Qu.:3599   3rd Qu.:3577   3rd Qu.:3632
## Max.   :5000   Max.   :4199   Max.   :4197   Max.   :4200
##       sd                var
## Min.   : 0.8756   Min.   :  0.7667
## 1st Qu.:15.9360   1st Qu.:253.9556
## Median :18.0012   Median :324.0444
## Mean   :17.6238   Mean   :322.5932
## 3rd Qu.:20.2989   3rd Qu.:412.0444
## Max.   :23.4132   Max.   :548.1778
```

```
head(df)
```

```
##   time   mean y_min y_max       sd       var
## 1    0 4198.9  4197  4200 0.875595 0.7666667
## 2    1 4197.9  4195  4200 1.523884 2.3222222
## 3    2 4197.4  4194  4199 1.712698 2.9333333
## 4    3 4196.4  4194  4199 1.577621 2.4888889
## 5    4 4195.6  4192  4198 1.955050 3.8222222
## 6    5 4194.4  4190  4197 2.170509 4.7111111
```

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 [%]"
)
```



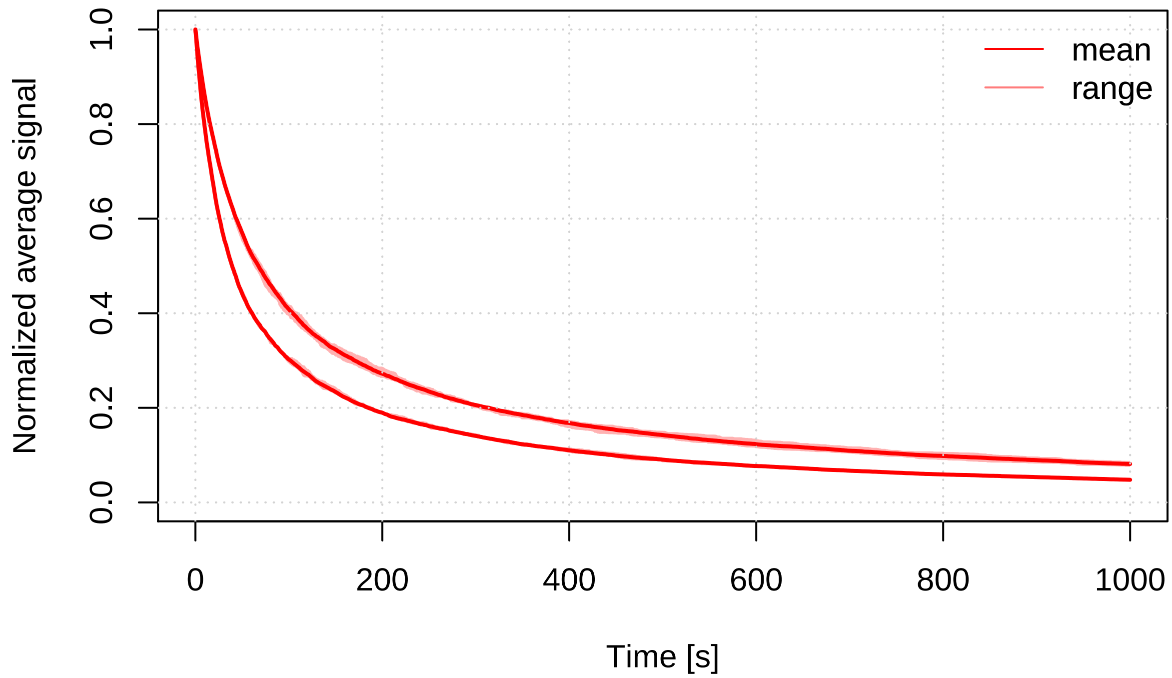
Example 2: Combining two plots

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)
```

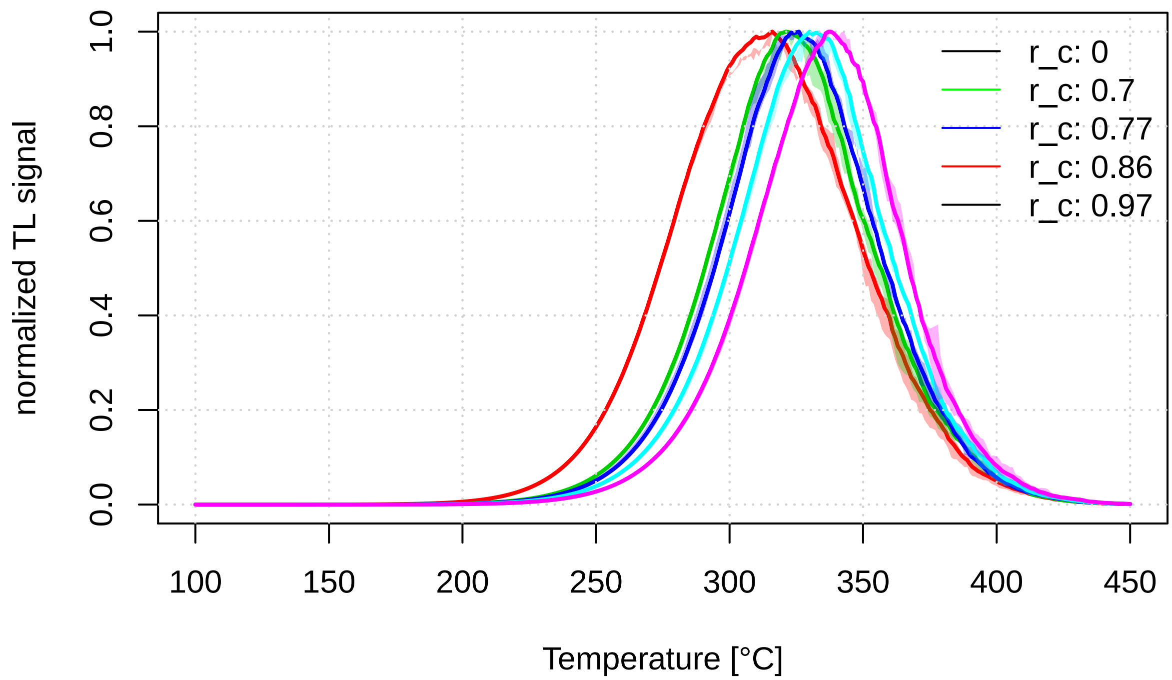


Example 3: Testing different parameters

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
  )

})
```


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