

RLumCarlo: Tedious features - fine examples

Sebastian Kreutzer, Johannes Friedrich, Vasilis Pagonis, Christoph Schmidt

RLumCarlo: v0.1.0.9000.137 / Last modified: 2019-11-27



1 Scope

RLumCarlo is a collection of energy-band models to simulate luminescence signals in dosimetric materials using Monte-Carlo (MC) methods for various stimulation modes. This document aims at supplementing the package documentation and it elaborates the package examples.

2 The models in RLumCarlo

The following table lists all 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	CORRESPONDING_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 is run by calling one of the **R** functions starting with `run_`. Currently, three different model types (TUN: tunnelling, 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.

3 RLumCarlo model parameters and variables

The following table summarises the parameters used in the implemented MC models along with their physical meaning, units and the range of realistic values. This range represents just a rough guideline and might be exceeded for particular cases.

Stimulation mode	Parameter	Parameter description	Unit	Realistic values
Delocalized TL	E	Thermal activation energy of the trap	eV	0.5–3
	s	Frequency factor of the trap	1/s	1E8–1E16
	times	Sequence of time steps for simulation	s	0–700
	clusters	Number of MC runs	1	1E1–1E4
	N_e	Total number of electron traps available	1	2–1E5
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1–1E5
Delocalized CW-IRSL	R	Delocalized retrapping ratio	1	0–1
	A	Optical excitation rate from trap to conduction band	1/s	1E–3–1
	times	Sequence of time steps for simulation	s	0–500
	clusters	Number of MC runs	1	1E1–1E4
	N_e	Total number of electron traps available	1	2–1E5
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1–1E5
Delocalized ISO	R	Delocalized retrapping ratio	1	0–1
	E	Thermal activation energy of the trap	eV	0.5–3
	s	Frequency factor of the trap	1/s	1E8–1E16
	T	Temperature	°C	20–300
	times	Sequence of time steps for simulation	s	0–1000
	clusters	Number of MC runs	1	1E1–1E4
Delocalized LM-OSL	N_e	Number of electrons	1	2–1E5
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1–1E5
	R	Delocalized retrapping ratio	1	0–1
	A	Optical excitation rate from trap to conduction band	1/s	1E–3–1
	times	Sequence of time steps for simulation	s	0–3000
	clusters	Number of MC runs	1	1E1–1E4
Localized TL	N_e	Total number of electron traps available	1	2–1E5
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1–1E5
	R	Delocalized retrapping ratio	1	0–1
	E	Thermal activation energy of the trap	eV	0.5–3
	s	Frequency factor of the trap	1/s	1E8–1E16
	T	Temperature	°C	20–300

Localized CW-IRSL	s	Frequency factor of the trap	1/s	1E8–1E16
	times	Sequence of time steps for simulation	s	0–700
	clusters	Number of MC runs	1	1E1–1E4
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1–1E5
	r	Localized retrapping ratio	1	0–1E5
	A	Excitation rate from ground state of the trap to the excited state	1/s	1E–3–1
	times	Sequence of time steps for simulation	s	0–500
	clusters	Number of MC runs	1	1E1–1E4
Localized ISO	n_filled	Number of filled electron traps at the beginning of the simulation	1	1–1E5
	r	Localized retrapping ratio	1	0–1E5
	E	Thermal activation energy of the trap	eV	0.5–3
	s	Frequency factor of the trap	1/s	1E8–1E16
	T	Temperature	°C	20–300
	times	Sequence of time steps for simulation	s	0–1000
	clusters	Number of MC runs	1	1E1–1E4
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1–1E5
Localized LM-OSL	r	Localized retrapping ratio	1	0–1E5
	A	Excitation rate from ground state of the trap to the excited state	1/s	1E–3–1
	times	Sequence of time steps for simulation	s	0–3000
	clusters	Number of MC runs	1	1E1–1E4
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1–1E5
	r	Localized retrapping ratio	1	0–1E5
	E	Thermal activation energy of the trap	eV	0.5–3
	s	Frequency factor of the trap	1/s	1E8–1E16
TL with tunneling recombination	rho	Density of recombination centers (defined as ρ' in Huntley 2006)	1	1E–7–1E–4
	r_c	Critical distance (>0) that is to be inserted if the sample has been thermally and/or optically pretreated, so that the electron-hole pairs within r_c have already recombined	1	0–2
	times	Sequence of time steps for simulation	s	0–700
	clusters	Number of MC runs	1	1E1–1E4
	N_e	Total number of electron traps available	1	2–1E5
	delta.r	Increments of the unitless distance parameter r	1	1E–3–1E–1
	A	Excitation rate from ground state of the trap to the excited state	1/s	1E–3–1
	rho	Density of recombination centers (defined as ρ' in Huntley 2006)	1	1E–7–1E–4
CW-IRSL with tunneling recombination	times	Sequence of time steps for simulation	s	0–500
	clusters	Number of MC runs	1	1E1–1E4

ISO with tunneling recombination	N_e	Total number of electron traps available	1	2–1E5
	r_c	Critical distance (>0) that is to be inserted if the sample has been thermally and/or optically pretreated, so that the electron-hole pairs within r_c have already recombined	1	0–2
	delta.r	Increments of the unitless distance parameter r	1	1E-3–1E-1
	E	Thermal activation energy of the trap	eV	0.5–3
	s	Frequency factor of the trap	1/s	1E8–1E16
	T	Temperature	°C	20–300
	rho	Density of recombination centers (defined as ρ' in Huntley 2006)	1	1E-7–1E-4
	times	Sequence of time steps for simulation	s	0–1000
	clusters	Number of MC runs	1	1E1–1E4
	N_e	Total number of electron traps available	1	2–1E5
LM-OSL with tunneling recombination	r_c	Critical distance (>0) that is to be inserted if the sample has been thermally and/or optically pretreated, so that the electron-hole pairs within r_c have already recombined	1	0–2
	delta.r	Increments of the unitless distance parameter r	1	1E-3–1E-1
	A	Excitation rate from ground state of the trap to the excited state	1/s	1E-3–1
	rho	Density of recombination centers (defined as ρ' in Huntley 2006)	1	1E-7–1E-4
	times	Sequence of time steps for simulation	s	0–3000
	clusters	Number of MC runs	1	1E1–1E4
	N_e	Total number of electron traps available	1	2–1E5
	r_c	Critical distance (>0) that is to be inserted if the sample has been thermally and/or optically pretreated, so that the electron-hole pairs within r_c have already recombined	1	0–2
	delta.r	Increments of the unitless distance parameter r	1	1E-3–1E-1

4 Examples

The following examples illustrate the capacity of RLumCarlo, by using code-snippets deploying longer simulation times than allowed for the standard package examples, which aim at a functionality test.

4.1 Example 1: A first example

The first examples an iso-thermal curve using the tunnelling model (other models work similar). Returned are either the simulated signal or the estimated remaining charges. The Function `plot_RLumCarlo()` provides an easy way to visualise the modelling results and is here called using the tee operator `%T>` from the package `magrittr` (which is imported by RLumCarlo). Simulation results are stored in the object `results` while, at the same time, piped to the function `plot_RLumCarlo()` for the output visualisation.

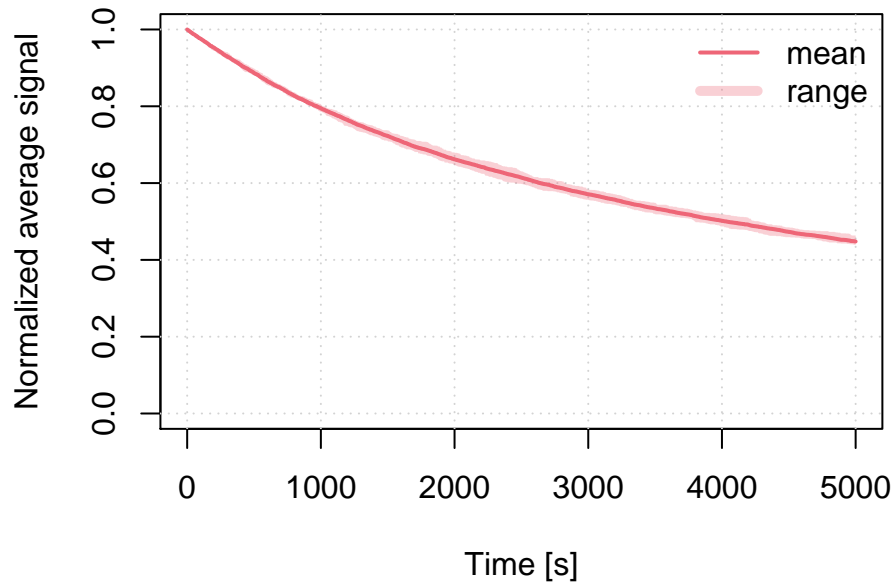
4.1.1 Model the signal

The most obvious modelling output is the luminescence signal itself, our example below simulates an iso-thermal (ITL) signal for a temperature (T) of 200 °C over 5,000 s using a tunnelling transition model. Trap parameters are $E = 1.2$ eV for the trap depth and a frequency factor for 1×10^{10} (1/s). The parameter `rho` (ρ) defines the recombination centre density.

```

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)

```



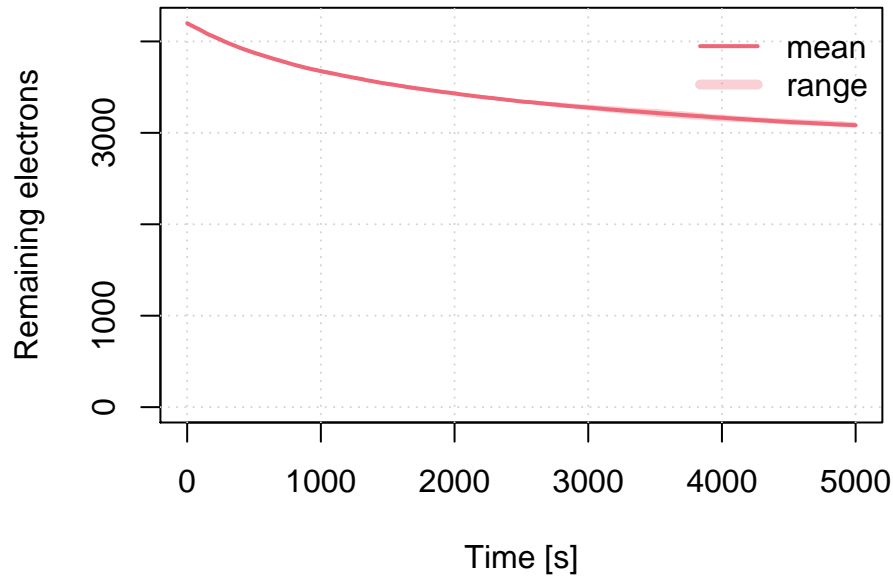
4.1.2 Model remaining charges

The first example can be slightly altered to provide alternative insight. Instead of the luminescence signal, below the variant returns the number of remaining electrons in the trap.

```

results <- run_MC_ISO_TUN(
  E = 1.2,
  s = 1e10,
  T = 200,
  rho = 0.007,
  times = seq(0, 5000),
  output = "remaining_e"
) %T>%
plot_RLumCarlo(
  legend = TRUE,
  ylab = "Remaining electrons"
)

```



4.1.3 Understanding the numerical output

In both cases the modelling output is an object of class `RLumCarlo_Model_Output`, which is basically a list consisting of an array and a numeric (vector).

```
str(results)
```

```
## List of 2
## $ signal: num [1:5001, 1:21, 1:10] 200 199 199 199 199 199 199 198 197 197 ...
##   ..- 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()` to simplify the data before there are plotted.

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

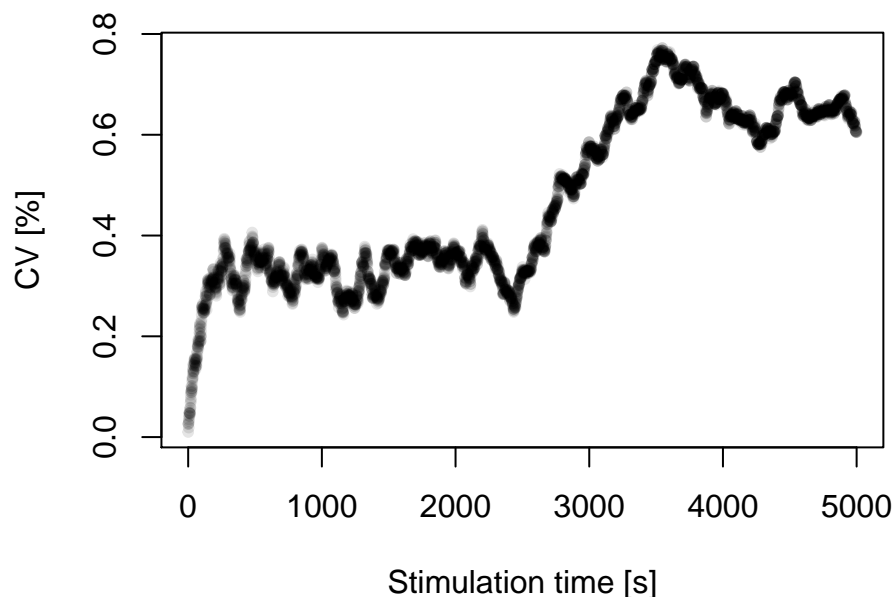
```
##      time      mean      y_min      y_max      sd
## Min.   : 0      Min.   :3083    Min.   :3058    Min.   :3118    Min.   : 0.4216
## 1st Qu.:1250    1st Qu.:3191    1st Qu.:3151    1st Qu.:3230    1st Qu.:11.9671
## Median :2500    Median :3343    Median :3323    Median :3358    Median :13.6561
## Mean   :2500    Mean   :3425    Mean   :3400    Mean   :3451    Mean   :15.6721
## 3rd Qu.:3750    3rd Qu.:3602    3rd Qu.:3587    3rd Qu.:3617    3rd Qu.:20.0953
## Max.   :5000    Max.   :4200    Max.   :4199    Max.   :4200    Max.   :24.8068
##      var
## Min.   : 0.1778
## 1st Qu.:143.2111
## Median :186.4889
## Mean   :267.7090
## 3rd Qu.:403.8222
## Max.   :615.3778
```

```
head(df)
```

```
##   time  mean y_min y_max      sd      var
## 1    0 4199.8  4199  4200 0.4216370 0.1777778
## 2    1 4198.6  4197  4200 1.0749677 1.1555556
## 3    2 4197.9  4196  4199 1.1005049 1.2111111
## 4    3 4197.5  4196  4199 1.1785113 1.3888889
## 5    4 4196.8  4196  4198 0.7888106 0.6222222
## 6    5 4196.0  4193  4198 1.3333333 1.7777778
```

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 against coefficient of variation (CV in %):

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



4.2 Example 2: Combining two plots

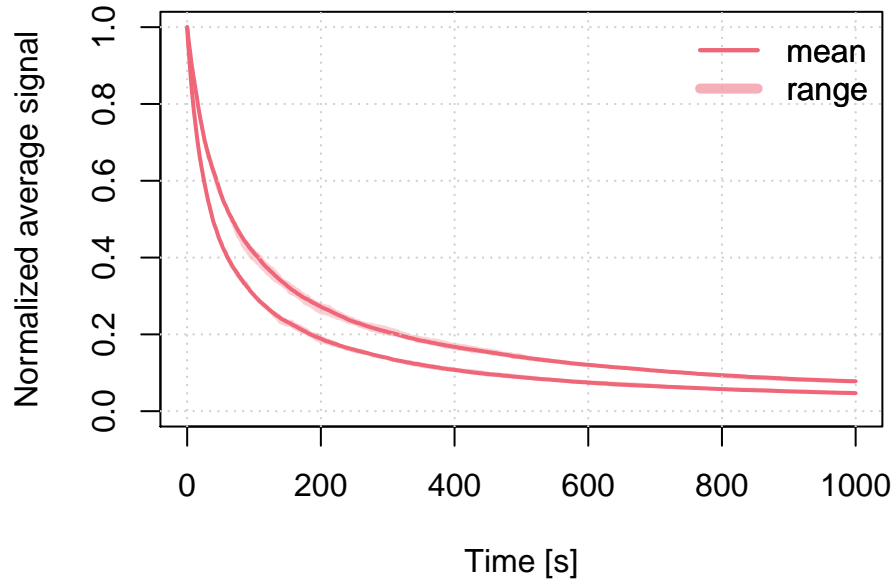
The following examples use again the tunnelling model but for continuous wave (CW) infrared light stimulation (IRSL), and they combine two plots in one single plot window.

```
## set time vector
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)
```



4.3 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 and it will then iterate automatically over the results to create a combined plot.

```
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) # here time = temperature
results <- lapply(r_c, function(x) {
  run_MC_TL_TUN(
    s = s,
    E = E,
    rho = rho,
    r_c = x,
    times = times
  )
})
```

The plot output can be highly customised to provide a better visual experience, e.g., the manual setting of the colours and the legend.

```
## plot curves, but without legend
plot_RLumCarlo(
  object = results,
  ylab = "normalised TL signal",
  xlab = "Temperature [°C]",
  plot_uncertainty = "range",
  col = khroma::colour("bright")(length(r_c)),
  legend = FALSE,
  norm = TRUE
```



```

)

## add legend manually
legend(
  "topleft",
  bty = "n",
  legend = paste0("r_c: ", r_c),
  lty = 1,
  col = khroma::colour("bright")(length(r_c))
)

```

