‘RLumCarlo’: Tedious features - fine examples

Sebastian Kreutzer, Johannes Friedrich, Vasilis Pagonis, Christoph Schmidt

RLumCarlo: v0.1.8 | last modified: 2022-02-20

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 elaborating the package examples.

2 The models in ‘RLumCarlo’

2.1 Overview

TRANSITION	BASE MODEL	IRSL	OSL	LM-OSL	TL
Delocalised	OTOR	-	X	X	X
Localised	GOT	X	-	X	X
Excited state tunnelling	LTM	X	-	X	X

In the table above column headers refer to stimulation modes, which are infrared stimulated luminescence (IRSL), optically stimulated luminescence (OSL), LM-OSL (Bulur 1996), and thermally stimulated luminescence (short: TL). In the column ‘BASE MODEL’ OTOR refers to ‘One Trap-One Recombination Centre’, GOT to ‘General One Trap’, and LTM to ‘Localized Transition Model’ (Jain, Guralnik, and Andersen 2012; Pagonis et al. 2019). For general overview we refer to the excellent book by Chen and Pagonis (2011).

2.2 Where to find them

The following table lists models as 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 (heating rate is 1 K/s)	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
	R	Delocalized retrapping ratio	1	0–1
Delocalized CW-IRSL	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
	R	Delocalized retrapping ratio	1	0–1
Delocalized ISO	E	Thermal activation energy of the trap	eV	0.5–3
	s	Frequency factor of the trap	1/s	1E8–1E16
	T	Temperature of the isothermal process	°C	20–300
	times	Sequence of time steps for simulation	s	0–1000
	clusters	Number of MC runs	1	1E1–1E4
	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
Delocalized LM-OSL	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
	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
Localized 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 (heating rate 1 K/s)	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
Localized CW-IRSL	A	Optical 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
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1–1E5
	r	Localized retrapping ratio	1	0–1E5
Localized ISO	E	Thermal activation energy of the trap	eV	0.5–3
	s	Frequency factor of the trap	1/s	1E8–1E16
	T	Temperature of the isothermal process	°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
	r	Localized retrapping ratio	1	0–1E5
Localized LM-OSL	A	Optical 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
TL with tunneling recombination	E	Thermal activation energy of the trap	eV	0.5–3
	s	Effective frequency factor of the tunneling process	1/s	1E8–1E16
	rho	Dimensionless 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 (heating rate 1 K/s)	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
CW-IRSL with tunneling recombination	A	Effective optical excitation rate of the tunneling process	1/s	1E-3–1
	rho	Dimensionless density of recombination centers (defined as ρ' in Huntley 2006)	1	1E-7–1E-4
	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
	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
ISO with tunneling recombination	E	Thermal activation energy of the trap	eV	0.5–3
	s	Effective frequency factor of the tunneling process	1/s	1E8–1E16
	T	Temperature of the isothermal process	°C	20–300
	rho	Dimensionless 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
	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
LM-OSL with tunneling recombination	A	Effective optical excitation rate of the tunneling process	1/s	1E-3–1
	rho	Dimensionless 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.

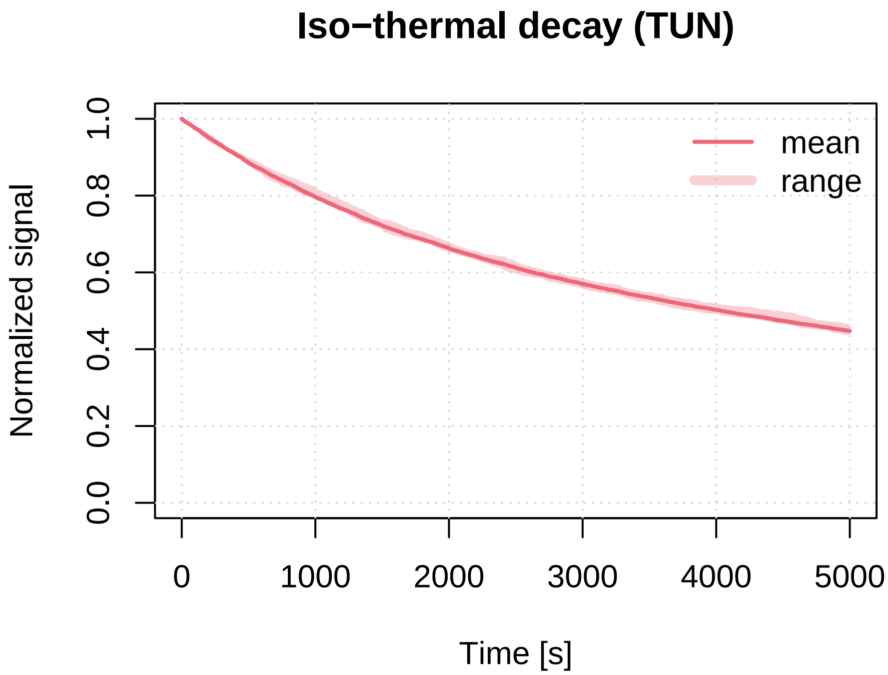
4.1 Example 1: A first example

The first example is an iso-thermal decay curve using the tunnelling model (other models work similarly). Returned are either the simulated signal or the estimated remaining trapped charge carriers. 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

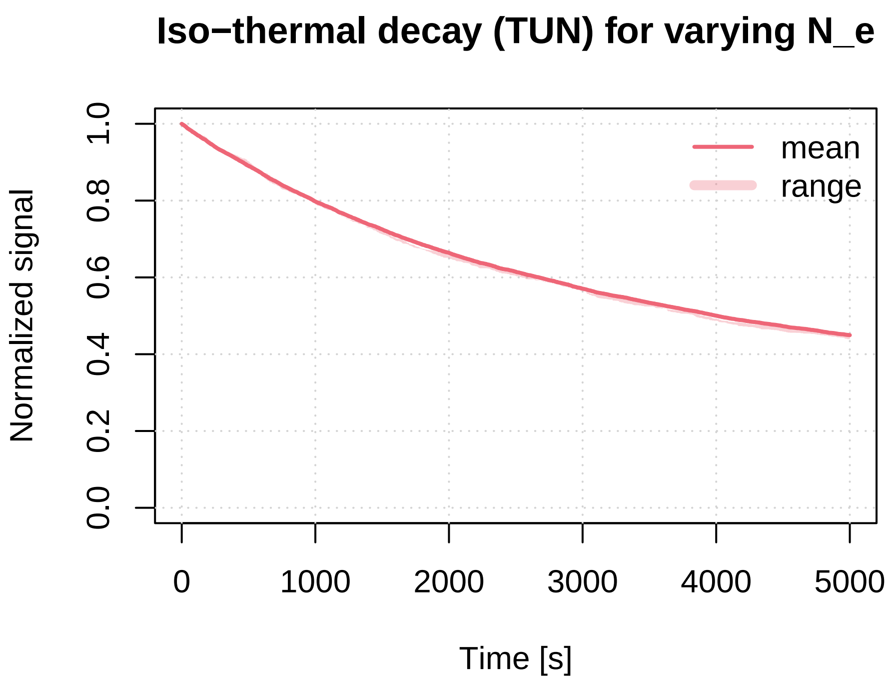
The most obvious modelling output is the luminescence signal itself, our example below simulates an iso-thermal (ITL) signal for a temperature (`T`) of 200 °C over 5,000 s using a tunnelling transition model. Trap parameters are $E = 1.2$ eV for the trap depth and a frequency factor of 1×10^{10} (1/s). The parameter `rho` (ρ') defines the recombination centre density.

```
results <- run_MC_ISO_TUN(
  E = 1.2,
  s = 1e10,
  T = 200,
  N_e = 200,
  rho = 0.007,
  clusters = 10,
  times = seq(0, 5000)
) %T>%
  plot_RLumCarlo(norm = TRUE,
                 legend = TRUE,
                 main = "Iso-thermal decay (TUN)")
```



In the example above `N_e` is a scalar, which means that all clusters start with the same number of electrons (here 200). However, 'RLumCarlo' supports different starting conditions with regard to the initial number of electrons. For example, one could assume that the number of initial electrons vary randomly between 190 and 210. Such a situation is created in the next example. Generally, 'RLumCarlo' supports such an input for the parameters `N_e` and `n_filled`.

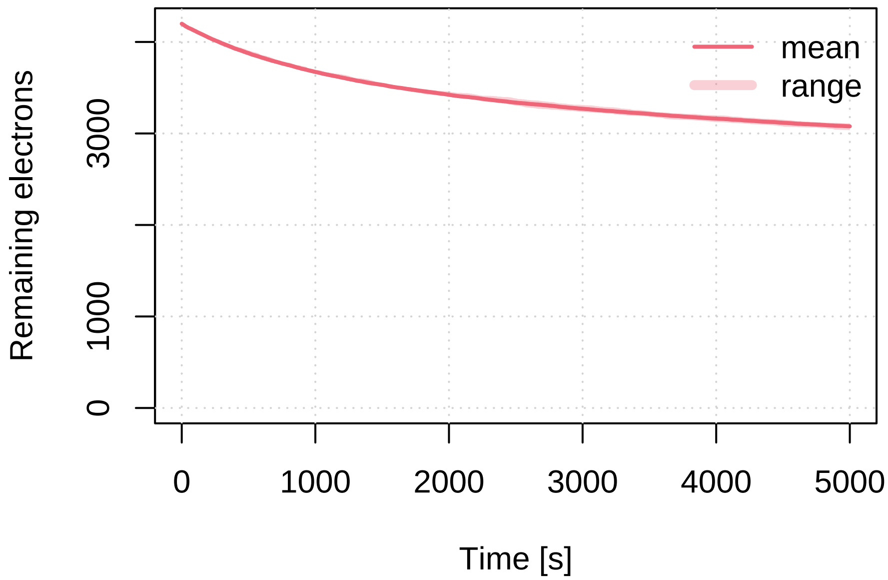
```
results <- run_MC_ISO_TUN(
  E = 1.2,
  s = 1e10,
  T = 200,
  N_e = sample(190:210,10,TRUE),
  rho = 0.007,
  clusters = 10,
  times = seq(0, 5000)
) %T>%
  plot_RLumCarlo(norm = TRUE,
                 legend = TRUE,
                 main = "Iso-thermal decay (TUN) for varying N_e")
```



4.1.2 Model remaining charges

The first example can be slightly altered to provide alternative insight. Instead of the luminescence signal, the variant below 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] 200 199 199 199 199 199 199 199 199 199 ...
##   ..- 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)
```

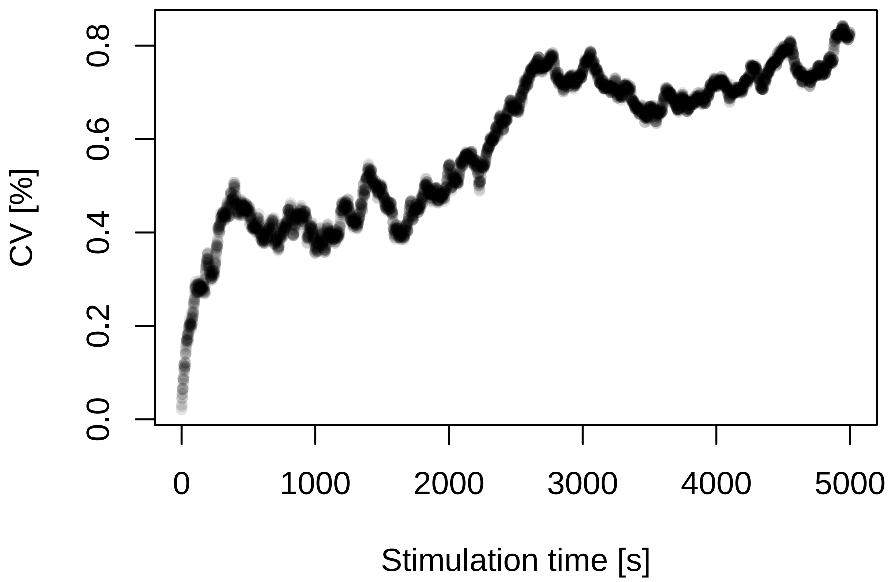
```
##       time           mean          y_min          y_max            sd
##  Min.   :   0   Min.   :3078   Min.   :3035   Min.   :3110   Min.   : 0.8756
##  1st Qu.:1250   1st Qu.:3186   1st Qu.:3149   1st Qu.:3217   1st Qu.:16.2344
##  Median :2500   Median :3337   Median :3305   Median :3383   Median :21.1618
##  Mean   :2500   Mean   :3421   Mean   :3392   Mean   :3453   Mean   :19.6248
##  3rd Qu.:3750   3rd Qu.:3596   3rd Qu.:3571   3rd Qu.:3630   3rd Qu.:23.0133
##  Max.   :5000   Max.   :4199   Max.   :4198   Max.   :4200   Max.   :25.9763
##       var                sum
##  Min.   :  0.7667   Min.   :30780
##  1st Qu.:263.5556   1st Qu.:31855
##  Median :447.8222   Median :33374
##  Mean   :402.8996   Mean   :34210
##  3rd Qu.:529.6111   3rd Qu.:35958
##  Max.   :674.7667   Max.   :41991
```

```
head(df)
```

```
##   time   mean y_min y_max       sd       var   sum
## 1    0 4199.1  4198  4200 0.875595 0.7666667 41991
## 2    1 4198.3  4196  4200 1.337494 1.7888889 41983
## 3    2 4197.7  4196  4199 1.159502 1.3444444 41977
## 4    3 4196.7  4194  4199 1.888562 3.5666667 41967
## 5    4 4196.0  4193  4198 1.885618 3.5555556 41960
## 6    5 4194.4  4189  4197 2.270585 5.1555556 41944
```

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 [%]"
)
```



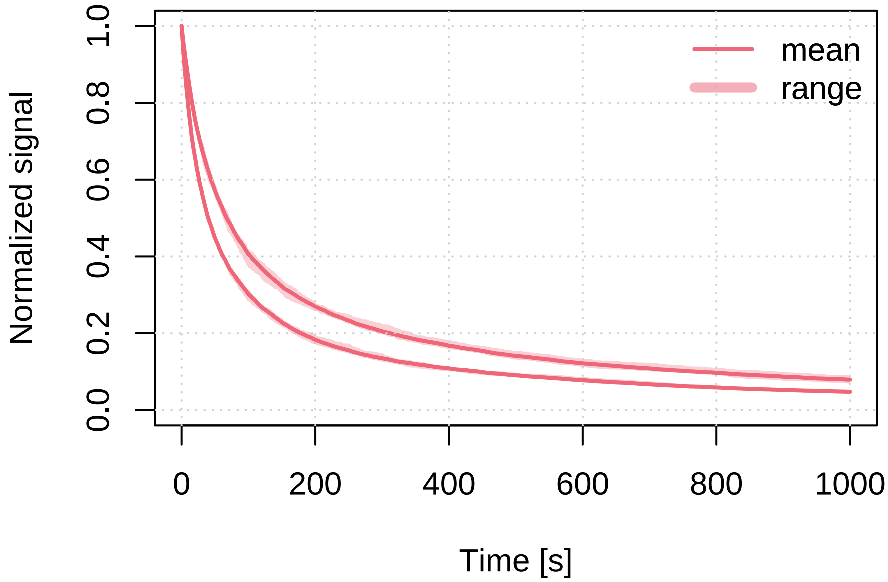
4.2 Example 2: Combining two plots

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.

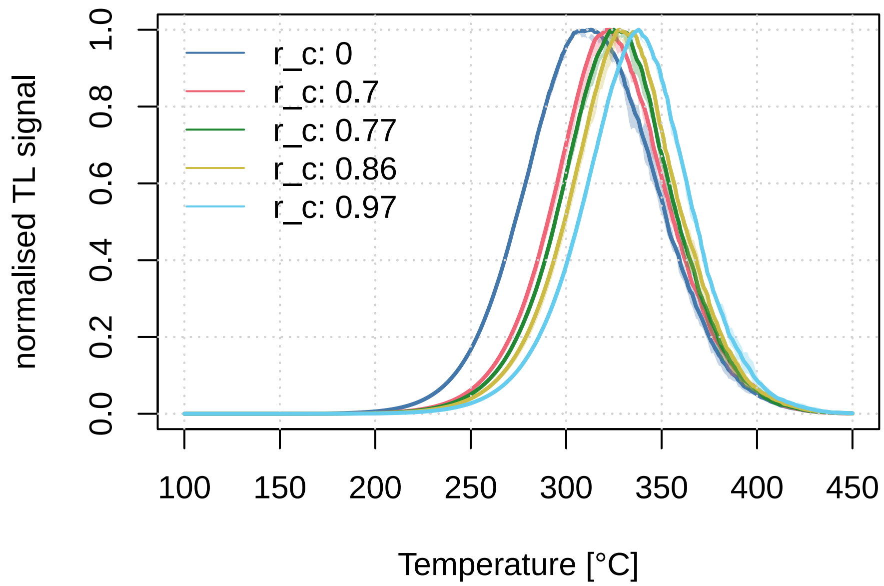
```
s <- 3.5e12
rho <- 0.015
E <- 1.45
r_c <- c(0,0.7,0.77,0.86, 0.97)
times <- seq(100, 450) # here time = temperature
results <- lapply(r_c, function(x) {
  run_MC_TL_TUN(
    s = s,
    E = E,
    rho = rho,
    r_c = x,
    times = times
  )

})
```

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 [\u00b0C]",
  plot_uncertainty = "range",
  col = khroma::colour("bright")(length(r_c)),
  legend = FALSE,
  norm = TRUE
)
```

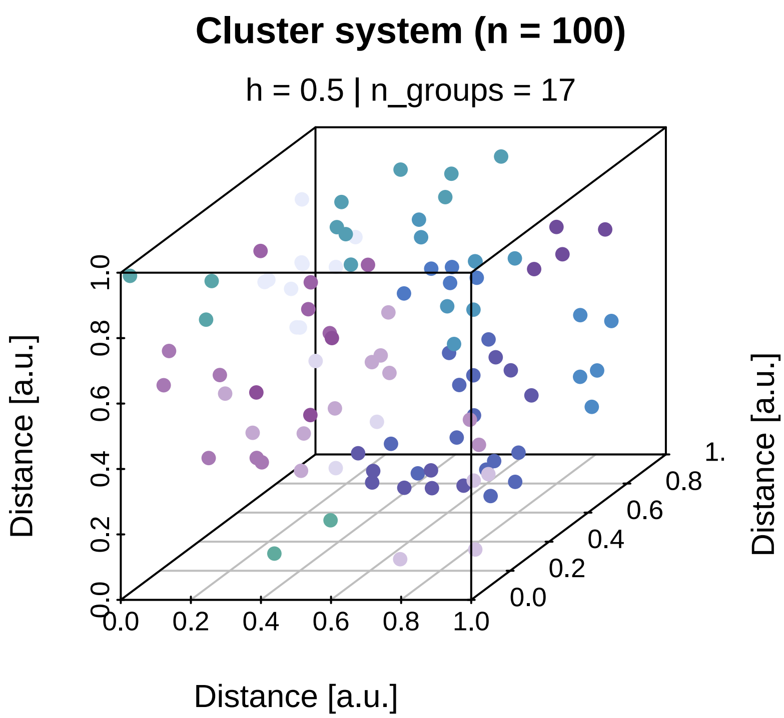
```r
## add legend manually
legend(
  "topleft",
  bty = "n",
  legend = paste0("r_c: ", r_c),
  lty = 1,
  col = khroma::colour("bright")(length(r_c))
)
```



4.4 Example 4: Dosimetric cluster systems

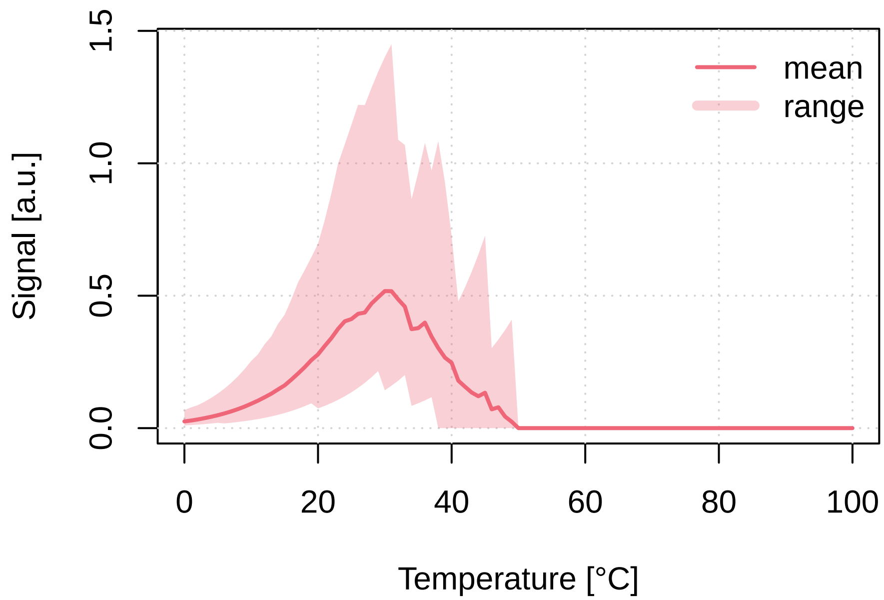
'RLumCarlo' supports the simulation of a cheap dosimetric cluster system with spatial correlation. Such a dosimetric cluster system can be created with the function `create_ClusterSystem()`:

```r
clusters <- create_ClusterSystem(n = 100, plot = TRUE)
```



The result is an arbitrary dosimetric system with randomly distributed clusters. The Euclidean distance is used to group the clusters (colour code). To use the system in the simulation, instead of providing a scalar as input to `clusters`, the output of `create_ClusterSystem()` can be injected in every `run_MC` function.

```
run_MC_TL_LOC(
 s = 1e14,
 E = 0.9,
 times = 0:100,
 b = 1,
 n_filled = 1000,
 method = "seq",
 clusters = clusters,
 r = 1) %>%
plot_RLumCarlo()
```



Please note: For the simulation of a dosimetric cluster system, the meaning of `n_filled` changes. Instead of defining the number of electrons per cluster, it becomes the total number of electrons in the system. Electrons are distributed according to the grouping of the single clusters (the colours in the three-dimensional scatter plot). Within one group, electrons are distributed evenly.

References

Bulur, Enver. 1996. "An Alternative Technique For Optically Stimulated Luminescence (OSL) Experiment." *Radiation Measurements* 26 (5): 701–9. https://doi.org/10.1016/S1350-4487(97)82884-3.

Chen, R, and Vasilis Pagonis. 2011. *Thermally and Optically Stimulated Luminescence - A Simulation Approach*. Thermally and Optically Stimulated Luminescence a Simulation Approach. John Wiley & Sons, Ltd.

Jain, Mayank, Benny Guralnik, and Martin Thalbitzer Andersen. 2012. "Stimulated luminescence emission from localized recombination in randomly distributed defects." *Journal of Physics: Condensed Matter* 24 (38): 385402. https://doi.org/10.1088/0953-8984/24/38/385402.

Pagonis, Vasilis, Johannes Friedrich, Michael Discher, Anna Müller-Kirschbaum, Veronika Schlosser, Sebastian Kreutzer, Reuven Chen, and Christoph Schmidt. 2019. "Excited state luminescence signals from a random distribution of defects_ A new Monte Carlo simulation approach for feldspar." *Journal of Luminescence* 207: 266–72. https://doi.org/10.1016/j.jlumin.2018.11.024.