Tutorial analyse_baSAR()

Norbert Mercier and Sebastian Kreutzer

22 September, 2016

(package version 0.6.4 | `developers@r-luminescence.de`)

This tuorial is part of the supplementary data of the article: Mercier, N., Kreutzer, S., Christophe, C., Guérin, G., Guibert, P., Lahaye, C., Lanos, P., Philippe, A., Tribolo, C., 2016. Bayesian statistics in luminescence dating: The 'baSAR'-model and its implementation in the R package 'Luminescence', Ancient TL

Paulette and Micha

Imagine that our two colleagues Micha and Paulette would like to use the function `analyse_baSAR()`. Paulette is an experienced **R** user, and wants to use only **R** for her data analysis, since she knows about the advantage it has over other solutions. Conversely, Micha ever since loved *MS Excel*^{TM} because it comes with a graphical user interface and it is easy to use. He enjoys putting a lot of different information in cells, order these data,save them and immediately gets a global view on a single sheet of all the experiments he did on his sample.

Paulette: experienced **R** user
Micha: **R** beginner

Moreover, Micha is used to analyse data with the software *Analyst* (Duller, 2015), and usually just copy and paste the summary table provided by *Analyst* into *MS Excel*^{TM} sheets after he had analysed automatically a BIN-file. Micha will probably never start loving **R**, but as Paulette, he is convinced by the new possibilities Bayesian statistics offers.

For this tutorial we put ourselves in the position of Paulette and Micha to sketch the different ways the function `analyse_baSAR()` offers to analyse data. 0

Some notes before we can start ...

With this supplement one, real on a Risø TL/OSL DA-20 reader measured, single grain dataset comes delivered as two separate files:

1. `Supplement_Mercier_et_al_2016.binx` and
2. `Supplement_Mercier_et_al_2016_short.binx`.

However, as a watched pot may never boil, we will not look too closely to the data, but take them as what they are for us: example data to sketch the possibilities of the function `analyse_baSAR()`. Furthermore, to reduce the calculation time needed, we will only work with the 2nd (shortened) dataset, but the full data set may be used for an own analyses.

Our setup...
R version: 3.3.1
RStudio^{TM} version: 0.99.903

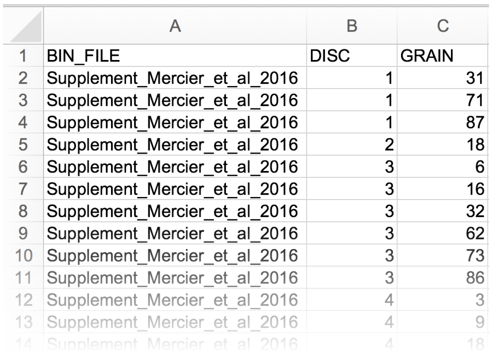
To precisely reproduce the scripts as given below all files (BIN-files, XLS-files) needed to be placed in the same folder as the **R** script file where the code is written in. Furthermore the working directory should be set (type `?setwd` in your **R** terminal) to the folder containing all these files.\footnote{You may consider creating a project in *RStudio*^{TM}

This tutorial was created using **R** Markdown (Allaire et al., 2015) and all presented **R** code snippets had been evaluated automatically while this PDF was created. Thus, if you are not able to reproduce the shown examples due to an error in **R**, its is likely a mistake on your side.

To simplify your life and easier following up this tutorial we recommend the free **R** programming environment *RStudio*^{TM} (`http://www.rstudio.com`).

STEP 1: - Get started

Paulette and Micha start with a similar set of input data, which is the shortened BIN-file comprising SAR single grain measurement data. Micha already inspected the dataset using the *Analyst*and selected only aliquots he felts comfortable with. His selection based on objective and scientifically well justified criteria and he ended up with an *MS Excel*^{TM} (Fig.1) table storing positions and grains he wants to use for the Bayesian analysis.



	A	B	C
1	BIN_FILE	DISC	GRAIN
2	Supplement_Mercier_et_al_2016	1	31
3	Supplement_Mercier_et_al_2016	1	71
4	Supplement_Mercier_et_al_2016	1	87
5	Supplement_Mercier_et_al_2016	2	18
6	Supplement_Mercier_et_al_2016	3	6
7	Supplement_Mercier_et_al_2016	3	16
8	Supplement_Mercier_et_al_2016	3	32
9	Supplement_Mercier_et_al_2016	3	62
10	Supplement_Mercier_et_al_2016	3	73
11	Supplement_Mercier_et_al_2016	3	86
12	Supplement_Mercier_et_al_2016	4	3
13	Supplement_Mercier_et_al_2016	4	9
14	Supplement_Mercier_et_al_2016	4	18

Figure 1: Screenshot of the XLS-file with the position and grain selection

By contrast, Paulette just started **R** and imports the BINX-file to her **R** session for a first brief inspection.

```
library(Luminescence)

BIN_file <- read_BIN2R(file = "Supplement_Mercier_et_al_2016_short.binx",
    txtProgressBar = FALSE)

##
## [read_BIN2R()]
##   >> Supplement_Mercier_et_al_2016_short.binx
##   >> 8484 records have been read successfully!

BIN_file
```

```
##
## [Risoe.BINfileData object]
##
##   BIN/BINX version      6
##   File name:            Supplement_Mercier_et_al_2016
##   Object date:          080116, 090116, 100116, 110116
##   User:                 Default
##   System ID:            999
##   Overall records:      8484
##   Records type:         OSL   (n = 8400)
##                         TL    (n = 84)
##   Position range:       1 : 6
##   Grain range:          0 : 100
##   Run range:            1 : 7
##   Set range:            11 : 16
```

As Micha, Paulette is faced with the problem that some of the measured grains (aliquots) are rather dim (or exhibit no light at all) and needed to be removed for further analyses. Paulette briefly considers to inspect the curves manually and subsequently reduce the dataset using the function `subset()` but then remembers what she had read in the article by Mercier et al. (2016) and that the function `verify_SingleGrainData()` will take over the job if no XLS-file is provided. This sounds promising.

However, Paulette is rather suspicious about **R** code she did not develope by herself and consults the manual via `?verify_SingleGrainData` in her **R** terminal and further gives the function `verify_SingleGrainData()` a glance by piping her BIN-file data into the function.

```
## load R package Luminescence
library(Luminescence)

## run verification
BIN_file_verified <- verify_SingleGrainData(object = BIN_file,
    threshold = 10, cleanup_level = "aliquot",
    plot = TRUE, verbose = FALSE)

## show resulting table
head(BIN_file_verified$unique_pairs)
```

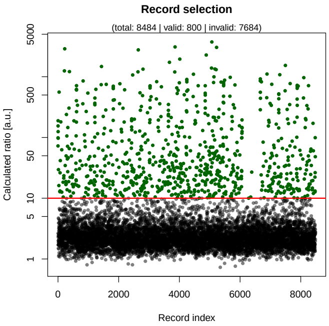


Figure 2: Graphical output of the single grain verification

```
##   POSITION GRAIN
## 1        1     0
## 2        2     0
## 3        3     0
## 4        4     0
## 5        5     0
## 6        6     0
```

We assume Paulette is satisfied for the moment and decides to allow the function to select aliquots automatically for her within the function `analyse_baSAR()`.

STEP 2: - A first 'baSAR' analysis

Now Paulette and Micha are ready for the first Bayesian analysis of their data in **R**. To recall the pre-conditions: Micha pre-selected his aliquots using the *Analyst*^{TM}} and provides his selection as XLS-file[1] Then, BIN-files with paths are compared with the list of BIN-files Micha provides in the object. The list of selected discs and associated grains are also read and stored in a list.} containing the list of selected discs / grains[2] he wants to include in the Bayesian calculation.

By contrast, Paulette did nothing so far and will rely on an automatic aliquot selection made within the function `analyse_baSAR()`

The function `analys_baSAR()` comes with a huge amount of parameters, allowing fine tuning and adjusting the analysis. Not all parameters needed to be set (optional) or come with an assumed to be useful, default value. In the first step we try to run an analysis by just setting the minimum of needed arguments.

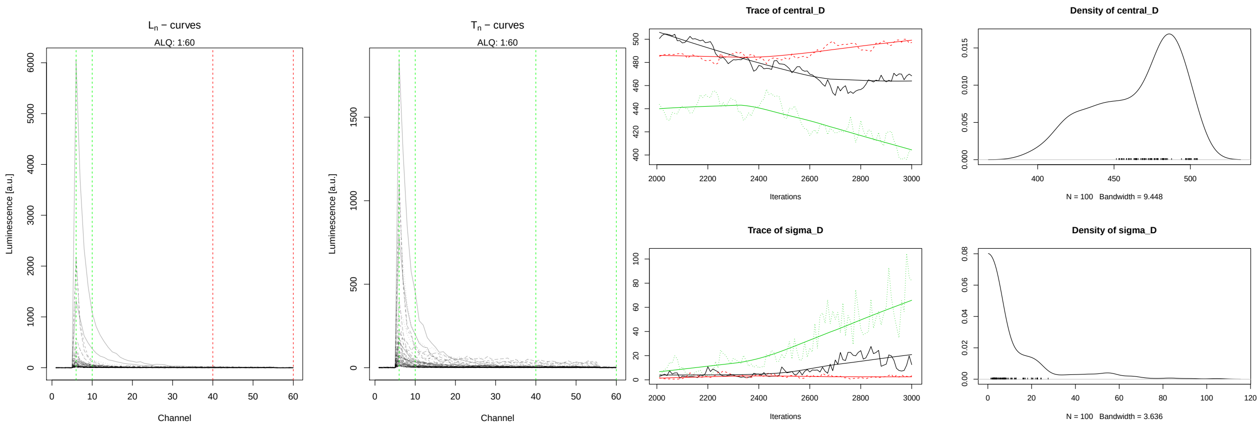
Before we can start, however, we need to load the **R** package 'Luminescence'.[3]

```
library(Luminescence)
```

Now we create our first function call with only required arguments. Micha also sets set the argument `XLS_file` as he provides his own grain selection.[4]. To reduce the computation time needed, we set the number of Monte Carlo Markov Chain runs to 1000 (argument `n.MCMC`). Please note that the argument `source_doserate` is required as the underlying models expect the dose values in Gy instead of s and thus the function will stop if no `source_doserate` is provided.

```
results <- analyse_baSAR(
  object = BIN_file,
  source_doserate = c(0.1, 0.001),
  XLS_file = "grain_selection.xls",
  signal.integral = c(6,10),
  background.integral = c(40, 60),
  n.MCMC = 1000


)
```

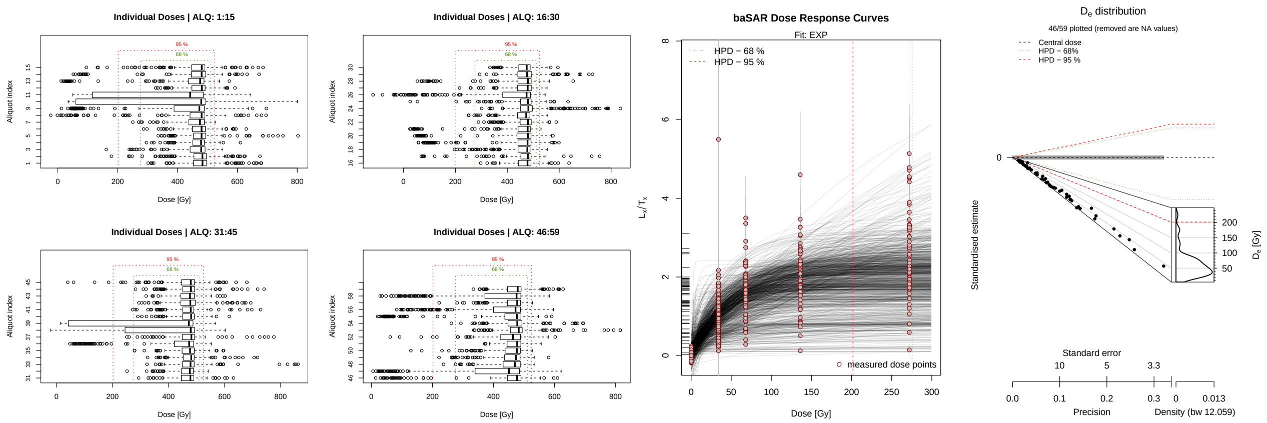


[1] The **R** code reads the XLS-file automatically. Empty rows are allowed and will be skipped during import. Only the first three columns are imported.

[2] One pair disc / grain pair defines one aliquot

[3] We will not load the package 'Luminescence' again in every code snippet, since, once loaded, we can assume that it remains attached to our **R** session

[4] For our example the grain selection table was produced using the function `verify_SingleGrainData`(), i.e. we except similar results



The (here minimised) graphical output of the function comprises five different plot types.[5]

1. The L_x/T_x curves from the data set, with indicated integral limits for the signal and background
2. The trace plots of the central dose (`central_D`) and the sigma of the central dose (`sigma_D`),
3. The corresponding density plots of `central_D` and `sigma_D`,
4. Boxplots showing the spread of the individual doses (on boxplot for each aliquot) together with the HPD at 68% and 95%
5. The individual dose response curves obtained during the Bayesian modelling plotted together with the L_x/T_x values
6. A plot (either an Abanico or KDE plot) with the D_e values obtained using the conventional approach and indicated the calculated central dose and the HPD

The also function accepts several parameters for growth curve fitting (`fit.method`, `fit.force_through-origin` and `fit.includingRecyclingPoints`) that can be passed as well to the function along with the model (distribution) used for the Bayesian calculation and used to represent the distribution of D_e values.

With this we have passed the first successful function run, even the results are not meaningful as it can been seen from the trace plots, strongly indicating a need for more MCMC runs. To avoid a time consuming repetition of the pre-processing we can use the object (`results`) produced by the first function run and pass it again in the function. For this 2nd run we also change the used model from `cauchy` (default) to `normal` and increase the number of MCMC runs to 10000. The argument `verbose` enables the terminal output.

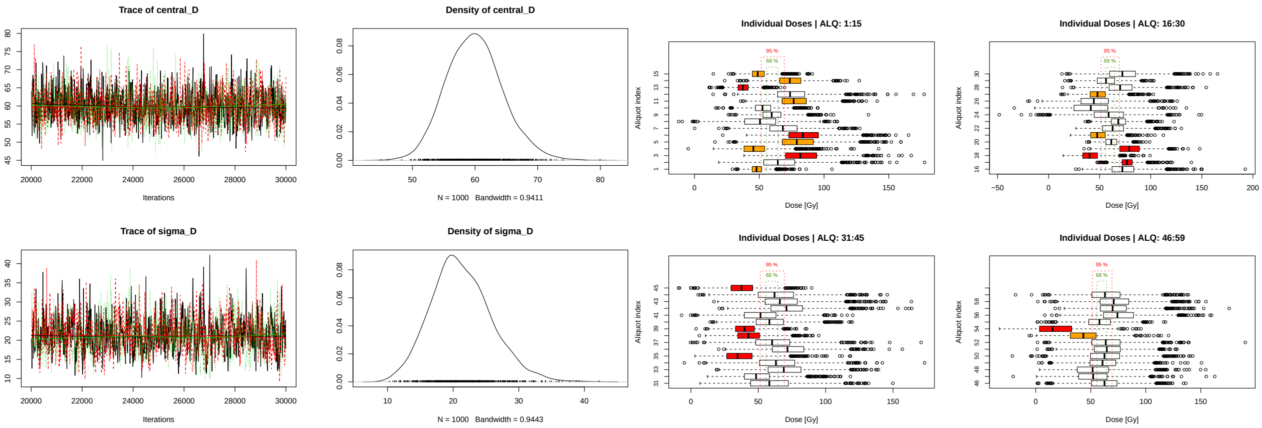
```
results_new <- analyse_baSAR(object = results,
    distribution = "normal", signal.integral = c(6,
        10), background.integral = c(30, 60),
    n.MCMC = 10000, verbose = TRUE)
```

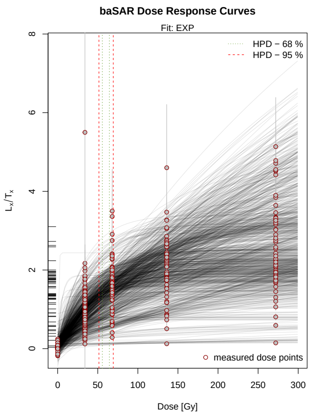
```
##
```

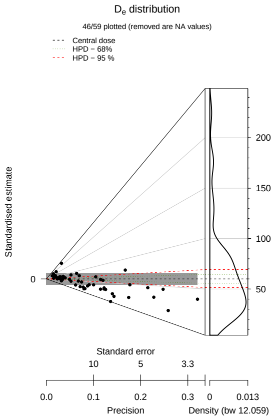
---

[5] If you run the function call on your computer, you will also see plots of the dose response curves obtained during the pre-processing

```
## [analyse_baSAR()] ---- baSAR-model ----
##
## +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
## [analyse_baSAR()] Bayesian analysis in progress ... Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 413
##    Unobserved stochastic nodes: 356
##    Total graph size: 11531
##
## Initializing model
##
## +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
##
##
## [analyse_baSAR()] ---- RESULTS ----
## ------------------------------------------------------------------
## Used distribution:       normal
## Number of aliquots used: 59/59
## Considered fitting method:   EXP
## Number of independent chains:    3
## Number MCMC iterations/chain:    10000
## ------------------------------------------------------------------
##              mean    sd  HPD
## >> Central dose:     60  4.514   [55.656 ; 64.461]**
##                      [51.447 ; 69.297]***
## >> sigma_D:          21.818  4.459   [17.464 ; 26.334]**
##                      [13.845 ; 31.294]***
## >> Final central De:     60  4.514    -   -
## ------------------------------------------------------------------
## (systematic error contribution to final De: 2.453844e-06 %)
## ** 68 % level | *** 95 % level
```







Apart from the terminal output the function returns all necessary values to understand and evaluate the calculation process. In the example the output is stored in the object `results_new` and containing five objects.

```
results_new
```

```
##
## [RLum.Results]
##  originator: analyse_baSAR()
##  data: 5
##      .. $summary : data.frame
##  .. $mcmc : mcmc.list
##  .. $models : list
##  .. $input_object : data.frame
##  .. $removed_aliquots : NULL
##  additional info elements:  1
```

For example the object `summary` is a table (`data.frame`) with the summarised calculation results[6]

```
results_new$summary[, c("DISTRIBUTION", "DE_FINAL",
    "DE_FINAL.ERROR")]
```

```
##  DISTRIBUTION DE_FINAL DE_FINAL.ERROR
## 1       normal       60          4.514
```

The objects `$mcmc` and `models` contain details on the ‘rjags’ (Plummer et al., 2016) (*JAGS* receptively, Plummer (2003)) output and the models available in the function. The object `input_object` contains the table that had been used as input data in the previous call where the function output (object `results`) was used as new input. The table `removed_aliquots` containsthe table of aliquots that had been manually or automatically removed during the pre-processing. In the current example no aliquot was removed.

STEP 3 - advanced manipulation

For both, Micha and Paulette, the presented standard options might not sufficiently account for their provided dataset. In particular it

[6] For the example only three columns have been selected

does not account for measurement data that should be rejected for various technical reasons.

Micha and Paulette use the table produced during the pre-processing to evaluate the measurement data.

Micha, e.g., first exports the table produced during the pre-processing using the *R* function `write.table()` subsequently imports the results in *Excel*^{TM} for a deeper evaluation. Paulette stays in *R* and gives the data a glance via `View(results_new$input_object)`.

Both conclude that some of the aliquots appear odd for the following reasons[7]:

1. Some aliquots erroneously comprise less than 7 cycles. These data were produced due to an error during the measurement and should be removed,
2. L_x/T_x values below zero appear not meaningful
3. the same applies for D_e values[8]

They conclude that some of the aliquots appear odd and thus they wanted to try the calculation again with these aliquots removed. While Micha writes down the row number from the exported table, Paulette uses *R* to identify these aliquots:

```
##create vector of aliquots to reject
reject <- which(
       results_new$input_object$CYCLES_NB != 7 |
       is.na(results_new$input_object$DE) |
       results_new$input_object$LxTx_3 < 0 |
       results_new$input_object$DE < 0


)

###show aliquots that should be
###rejected
reject

## [1]  3  5 24 34 43 44
```

Micha can create a similar vector by typing

```
reject <- c(3, 5, 24, 34, 43, 44)
```

In our case Micha and Paulette made a similar selection and pass this selection to the new function call using the argument `aliquot_range`. Please note that the leading `-` removes the aliquots wanted to be rejected. Alternatively a positive selection can be passed, i.e. a vector containing all aliquots wanted for the analysis.

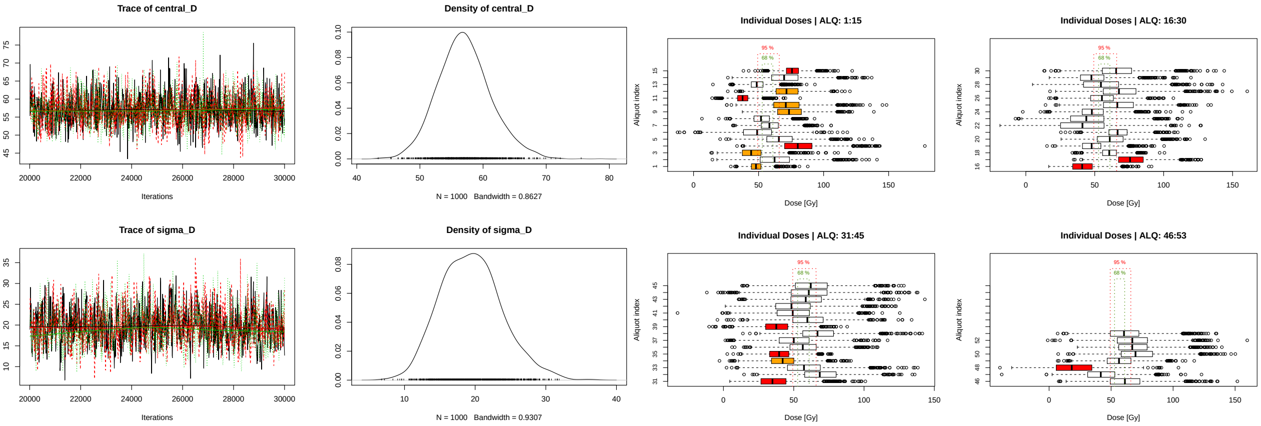
```
results_new2 <- analyse_baSAR(object = results,
    aliquot_range = -reject, distribution = "normal",
    signal.integral = c(6, 10), background.integral = c(30,
        60), n.MCMC = 10000, verbose = TRUE)
```

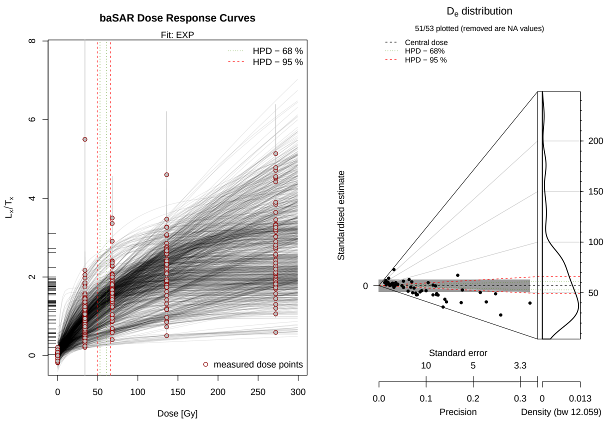
---

[7] The here mentioned criteria were produced with regard to the full data set, the short data set does not include aliquots comprising less than 7 cycles.

[8] Based on the pre-processing data should not rejected lightly, e.g., an 'NaN' D_e (not the here not selected 'NA') value usually indicates that the dose response curve fitting failed. This gives no reason to reject the aliquot for the Bayesian calculation.

```
##
## [analyse_baSAR()] ---- baSAR-model ----
##
## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
## [analyse_baSAR()] Bayesian analysis in progress ... Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 371
##    Unobserved stochastic nodes: 320
##    Total graph size: 10362
##
## Initializing model
##
## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
##
##
## [analyse_baSAR()] ---- RESULTS ----
## ------------------------------------------------------------------
## Used distribution:       normal
## Number of aliquots used: 53/59 (manually removed: 6)
## Considered fitting method:   EXP
## Number of independent chains:    3
## Number MCMC iterations/chain:    10000
## ------------------------------------------------------------------
##              mean    sd  HPD
## >> Central dose:     56.979  4.256   [52.789 ; 60.973]**
##                      [49.298 ; 65.897]***
## >> sigma_D:          19.732  4.278   [15.496 ; 23.95]**
##                      [11.964 ; 28.952]***
## >> Final central De:     56.979  4.256    -   -
## ------------------------------------------------------------------
## (systematic error contribution to final De: 2.760367e-06 %)
## ** 68 % level | *** 95 % level
```





The removed aliquots can now be found in the output object `removed_aliquots` which is now no longer empty.[9]

```
## show removed aliquots show only five columns
results_new2$removed_aliquots[, c("DISC", "GRAIN",
    "DE", "DE.SD", "D0")]
```

```
##    DISC GRAIN  DE DE.SD    D0
## 3     1    87 NaN    NA  1.00
## 5     3     6 NaN    NA  1.00
## 24    1    97  NA    NA    NA
## 34    5    64 NaN 15.68 12.43
## 43    4    95 NaN    NA  1.00
## 44    5     8 NaN    NA  1.00
```

Further selections might be undertaken with regard to the dose response curves created during the pre-processing. Nevertheless, it should be taken into account that any unjustified selection (e.g., all data that are not attributed as measurement error) may considerable bias the Bayesian calculation.

However, Paulette is not only an advanced *R* user, but also an expert in Bayesian statistics and she is till not happy with the results and doubt whether the underlying models are sufficient at all. She therefore decided to pass an own, modified model to the function, which gives her an ultimate control over the calculation. However, new models should at least follow the nomenclature set by the predefined models. The predefined models are part of the function output:

```
## show returned models
str(results_new2$models, nchar.max = 25)
```

```
## List of 4
##  $ cauchy      : chr "model {\"| __truncated__
##  $ normal      : chr "model {\"| __truncated__
##  $ log_normal  : chr "model {\"| __truncated__
##  $ user_defined: NULL
```

This call reveals that the function already considers the possibility of self-defined model and the `NULL` indicates that nothing is provided

[9] Please note again that the D_e values in the table are not(!) created during the Bayesian calculation, but the pre-processing.

so far. Paulette now decides to store one of the old models in a text file for modifications.

```
writeLines(text = results_new2$models$normal,
    con = file("myModel.txt"))
unlink("myModel.txt")
```

After the modifications were applied the model can piped as new model into the function call:

```
results_new3 <- analyse_baSAR(
  object = results,
  aliquot_range = -reject,
  baSAR_model = readLines(con = file("myModel.txt"))
  distribution = 'normal',
  signal.integral = c(6,10),
  background.integral = c(30, 60),
  n.MCMC = 10000,
  verbose = TRUE
)
```

References

Allaire, J., Horner, J., Marti, V., Porte, N., 2015. markdown: 'Markdown' Rendering for R.

Duller, G.A.T., 2015. The Analyst software package for luminescence data: overview and recent improvements. Ancient TL 33, 35–42.

Plummer, M., 2003. JAGS: A Program for Analysis of Bayesian Graphical Models using Gibbs Sampling. DSC 2003 Working Papers 124.

Plummer, M., Stukalov, A., Denwood, M., 2016. Interface to the JAGS MCMC library.