# Workshop

## **R**-package *Luminescence*

### Introduction into plot-functions

*Sebastian Kreutzer, Margret Fuchs, Michael Dietze, Manfred Fischer*

October 2012

Sebastian Kreutzer — Justus-Liebig-Universität Giessen
sebastian.kreutzer@geogr.uni-giessen.de

Margret Fuchs — TU Bergakademie Freiberg
fuchs@mailserver.tu-freiberg.de

Michael Dietze — TU Dresden
micha.dietze@mailbox.tu-dresden.de

Manfred Fischer — Universität Bayreuth
manfred.fischer@uni-bayreuth.de

Christoph Schmidt — Universität Bayreuth
christoph.schmidt@uni-bayreuth.de

Christoph Burow — Universität Köln
christoph.burow@uni-koeln.de

# Table of contents



**Annotations:**

Certain remarks are highlighted

***functions red***
***arguments green***
***variables blue***

R-commands will be displayed in a coloured frame

```
dev.off()
```

The output of an r-command will be displayed with a black frame

```
## null device
##           1
```

# 1. Plotting of a curve into a file

### 1.Step

The luminescence-package will need to be loaded.

```
library("Luminescence")
```

### 2. Step

With the help of the function ***readBIN2R*** a Risoe-Bin file will be loaded and assigned to the variable ***max***.
The only required argument is the path to the specified file.

```
max <- readBIN2R("D:/R/Daten/test2.BIN")
```

When you call the variable ***max***, its content will be displayed.

```
max
```

```
## Risoe.BINfileData Object
##  Version:              03
##  Object Date:          200120
##  User:                 Default
##  System ID:            0
##  Overall Records:      440
##  Records Type:         OSL=280; TL=140; IRSL=20;
##  Position Range:       1 : 20
##  Run Range:            2 : 43
##  Set Range:            1 : 2
```

### 3. Step

R offers a multitude of graphic formats for export.
In this example the Shine-down- and TL-curves of the in step 2 imported data will be plotted, and exported into a pdf-file. For this purpose we will use the function ***pdf*** with three arguments:
The first one determines the filename and the path where the file will be stored. The second one, ***paper*** arranges the page size and the last one, ***height***, the height of the diagrams.

```
pdf(file = "D:/R/WorkingDirectory/Plot_2/CurveOutput_test2_g.pdf",
paper = "a4",  height = 11)
```

### 4a. Step

The call of ***par(mfrow = c(2, 1))*** splits the graphic device into two rows and one column, so that two diagrams per page can be plotted. The number of rows and columns are assigned by the argument ***mfrow***. The first data of ***mfrow*** determines the number of rows, the second one the number of columns.

```
par(mfrow = c(2, 1))
```

see Plot 4a

### 4b. Step

In this case the call of ***par(mfrow = c(3, 4))*** divides the device into four rows and three columns, so that twelve diagrams can be plotted on one single page.

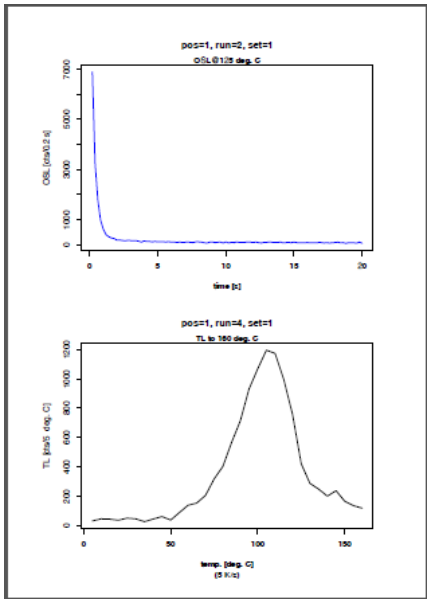
```
par(mfrow = c(3, 4))
```

see Plot 4b

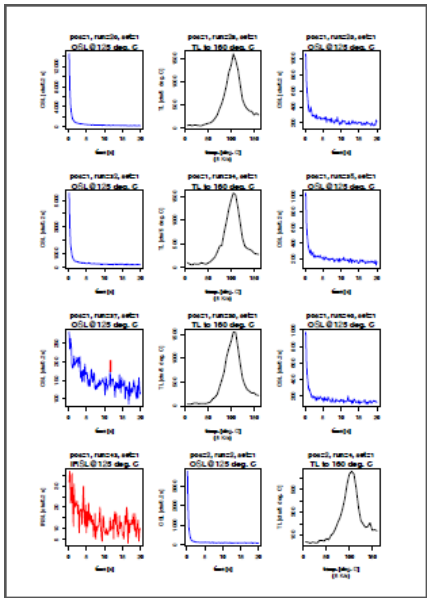
### 5. Step

The function ***plot_BINfileData*** will plot the individual diagrams.
The variable ***max***, in which the data is stored will be passed as argument.

```
plot_BINfileData(BINfileData = max)
```



Plot 4a



Plot 4b

### 6. Step

With the call of ***dev.off()*** the graphic device of the R-IDE will be closed.
This is a mandatory command, otherwise a blank document will be created.

```
dev.off()
```

```
## null device
##           1
```

## 2. Plot of a Shine-down-curve

### 1. Step

The first data set [1], of the variable ***max*** is called.
This record is of data type "list".

```
max@DATA[1]
```

```
## [[1]]
##  [1] 6891 3255 1792 984  624 401  320  265  245  180  185  169  152  171
## [15]  164  151  162  138  102 151  126  128  110  130  111  120  116  107
## [29]  119  100  106   86  104  87   96  106   93   83  112  111  102   79
## [43]   65   97  109   82   96  101   69  110   69   94   82  109   86   87
## [57]   96  101   99   88  103   83   68   95   93  108   95   95   84  106
## [71]   72   83   89   81  101   84   81   84   82   85   72   71   79   85
## [85]   85   68   79   67   75   87  100   75   85   51   78   71   73   64
## [99]  100   59
```

### 2. Step

Transformation of data type and variable assignment.
The list will be transformed into a vector, by the help of the function ***unlist***. The new created vector is assigned to the variable ***y***

```
y <- unlist(max@DATA[1])
```

### 3. Step

The first data set of ***max*** (max@METADATA) is assigned to the variable ***zeit***. The argument ***HIGH*** specifies the time range of the measurement.

```
zeit <- max@METADATA[which(max@METADATA[, "ID"] == 1), "HIGH"]
```

The call of ***zeit*** shows its value, in this case = 20 (20 sec).

```
zeit
```

```
## [1] 20
```

### 4. Step

The function ***length*** queries the length (the number of elements) of the vector, and writes the result to the variable ***laenge***.

```
laenge <- length(y)
```

The call of ***laenge*** shows the number of its elements, in this case =100.

```
laenge
```

```
## [1] 100
```

### 5. Step

The function ***seq*** creates a sequence of numbers with a user-defined interval. The initial value is assigned by the first argument (***zeit/laenge***, i.e. 20/100 == 0.2), the final value by the second argument, and the last one defines the interval between the individual values. The result of the function call will be stored in the variable ***x***.

```
x <- seq(zeit/laenge, zeit, by = zeit/laenge)
```

The call of ***x*** visualizes the sequence (the number of elements).

```
x
```

```
##  [1]  0.2  0.4  0.6  0.8  1.0  1.2  1.4  1.6  1.8  2.0  2.2  2.4  2.6  2.8
## [15]  3.0  3.2  3.4  3.6  3.8  4.0  4.2  4.4  4.6  4.8  5.0  5.2  5.4  5.6
## [29]  5.8  6.0  6.2  6.4  6.6  6.8  7.0  7.2  7.4  7.6  7.8  8.0  8.2  8.4
## [43]  8.6  8.8  9.0  9.2  9.4  9.6  9.8 10.0 10.2 10.4 10.6 10.8 11.0 11.2
## [57] 11.4 11.6 11.8 12.0 12.2 12.4 12.6 12.8 13.0 13.2 13.4 13.6 13.8 14.0
## [71] 14.2 14.4 14.6 14.8 15.0 15.2 15.4 15.6 15.8 16.0 16.2 16.4 16.6 16.8
## [85] 17.0 17.2 17.4 17.6 17.8 18.0 18.2 18.4 18.6 18.8 19.0 19.2 19.4 19.6
## [99] 19.8 20.0
```

### 6.Step

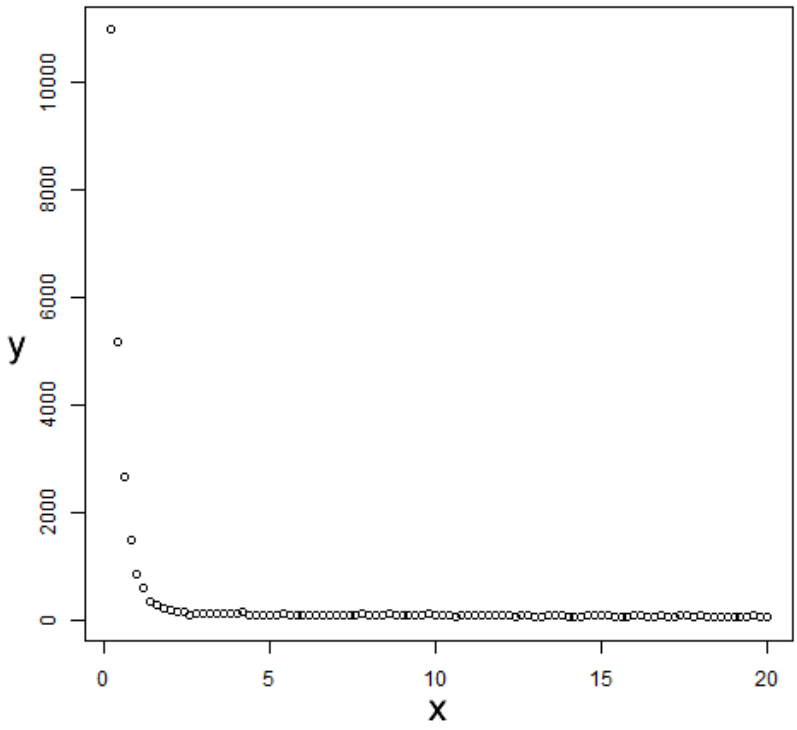
The variables ***x*** and ***y*** will be transformed into a "data.frame" and assigned to the variable ***xy***.

```
xy <- data.frame(x, y)
```

### 7. Step

In this step ***xy*** will be plotted with the help of the generic R-function ***plot***.

```
plot(xy)
```



# 3. The function *plot_GrowthCurve*

### 1.Schritt

The function ***Analyse_SAR.OSLdata*** will be called with the following arguments:
***max*** = Risoe.BINfileData
***c(1:2)*** = the integral from 1 to 2 is needed for analysis
***c(85:100)*** = the integral from 85 to 100 is substracted as background
The result is allocated to the variable ***z***.

```
z <- Analyse_SAR.OSLdata(max, c(1:2), c(85:100))
```

```
## [Analyse_OSLCurves.R] >> Position 51 is not valid and has been omitted!
```

### 2. Step

The first data set from the variable ***z*** will be extracted and assigned to the variable ***z1***.

```
z.1 <- z$LnLxTnTx[1]
```

### 3. Step

With the help of the function as, the object ***z1*** will be transformed into a data.frame. In R, one can transform nearly any data type into another one with the use of the function ***as***.

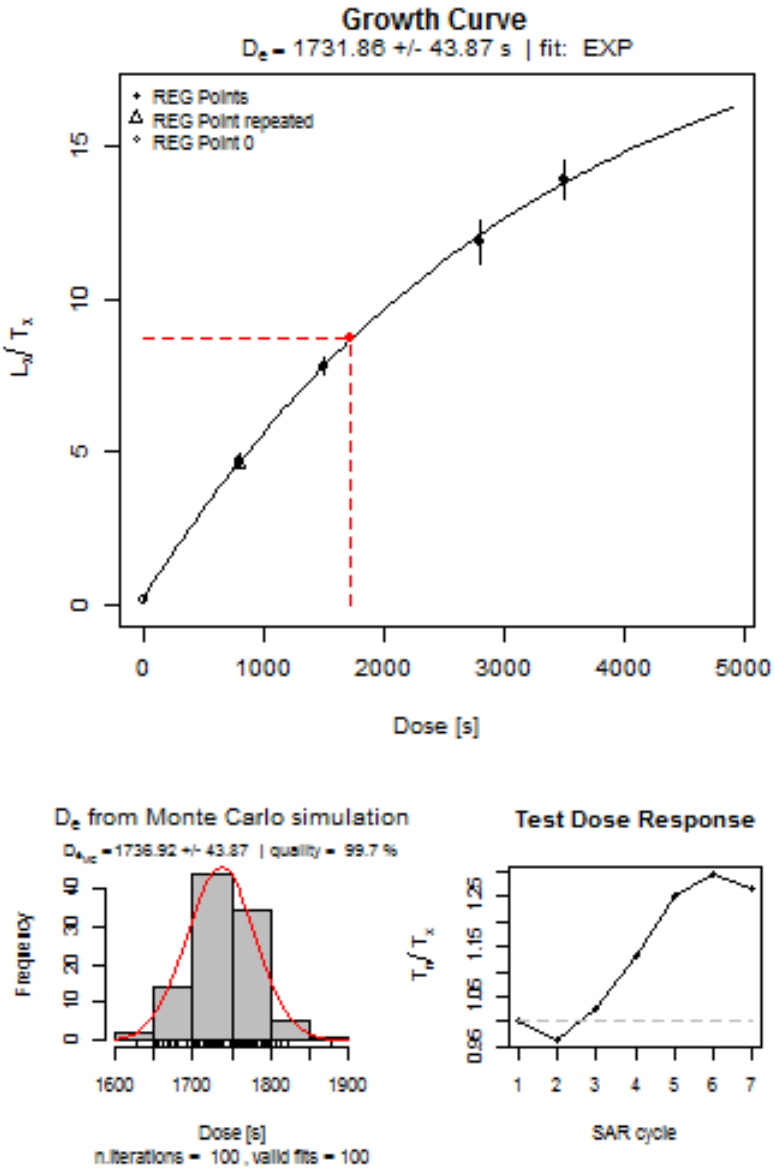
```
z.1 <- as.data.frame(z.1)
```

### 4. Step

In the last step the function ***plot_GrowthCurve*** is called. As the only neccesary argument the data set of ***z1*** is assigned.

```
plot_GrowthCurve(z.1[, c("Dose", "LxTx", "LxTx.Error", "TnTx")])
```

```
## [plot_GrowthCurve.R] >> D0 = 3253.27
```




```
## $De
##     De De.Error   D0
## 1 1732    43.87 3253
##
## $Fit
## Nonlinear regression model
##   model:  y ~ fit.functionEXP(a, b, c, x)
##    data:  data
##      a      b      c
##   20.8 3253.3   26.0
##  weighted residual sum-of-squares: 0.00249
##
## Algorithm "port", convergence message: relative convergence (4)
```

# 4. More plotfunctions:

## 4.1 plot_DeDistribution

## 4.2 plot_RadialPlot

## 4.3 plot_Histogram

### 1.Step: Data import

In the first step a csv-file is stored in the variable a with the help of the generic R-function ***read.csv***. The function will be called with three arguments. The first one shows the path to the csv-file, the next one - ***header*** - the labeling of the columns,and the last one - ***sep*** - defines a semicolon as separator between the individual data. By the call of ***a*** the content of the variable is displayed, and one can see the header of the two columns: ED and ED_Error.

```
a <- read.csv( "D:/R/Daten/MKQ.csv", header = TRUE, sep = ";")
```

The De distribution can now be plotted by the use of the function ***plot_DeDistribution***. The needed arguments are the data set of ***a***, as well as ***zlab*** with the parameters De and [s] to label the x-axis.

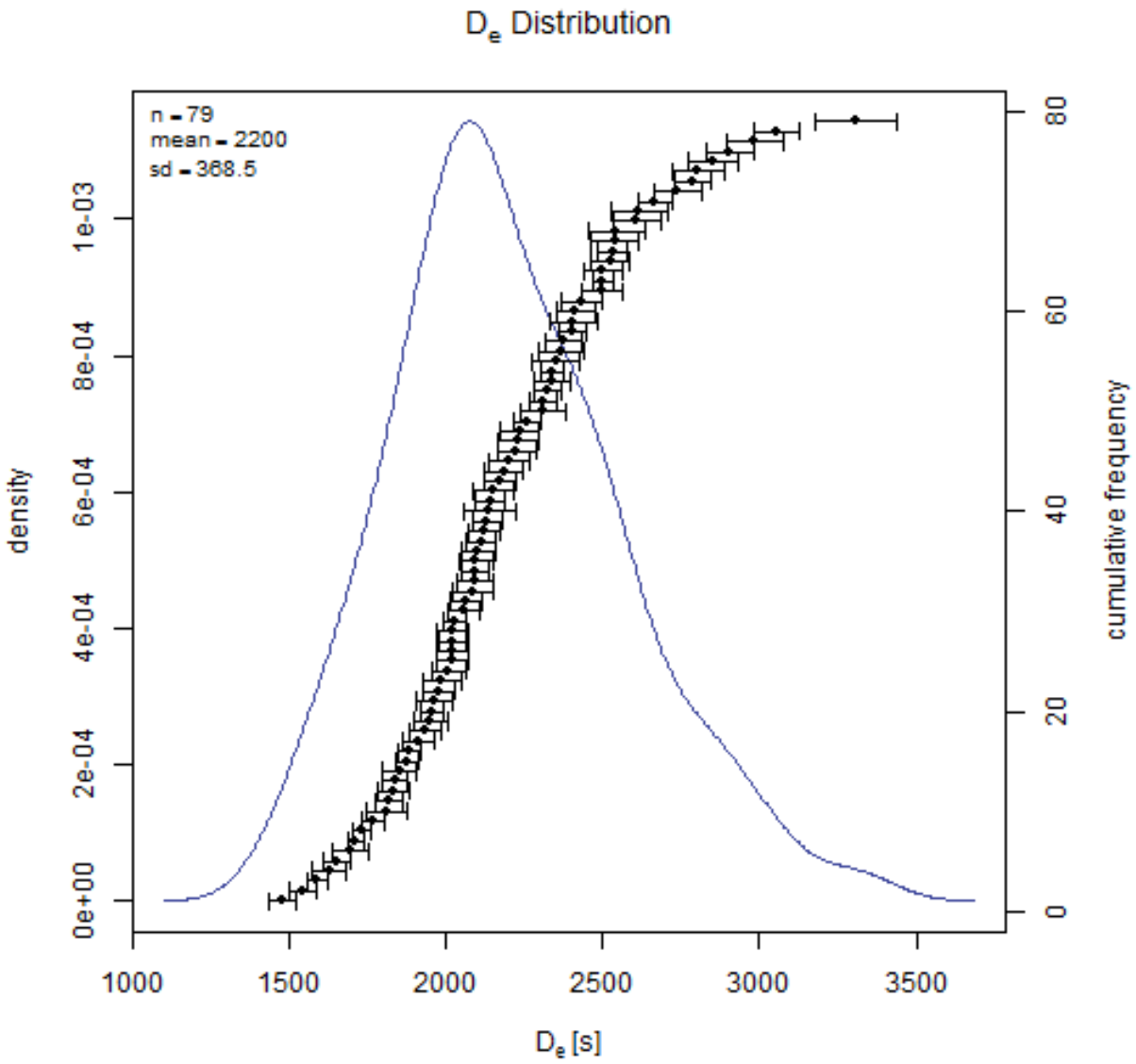
```
a
```

```
##       ED        ED_Error
## 1    1733      58.87
## 2    1913      98.80
## 3    1817      83.82
## 4    1884      67.08
## 5    2087      132.69
## 6    2313      87.26
## 7    2741      147.55
## 8    1543      91.72
## 9    2021      106.15
## 10   2852      159.21
## 11   2404      83.10
## 12   2133      89.96
## 13   1590      65.10
## 14   1696      111.37
## 15   2102      106.12
## etc.
```

## 2. Step: Plotting the De-distribution

The De distribution can now be plotted by the use of the function ***plot_DeDistribution***.
The needed arguments are the data set of ***a***, as well as ***zlab*** with the parameters De and [s] to label the x-axis.

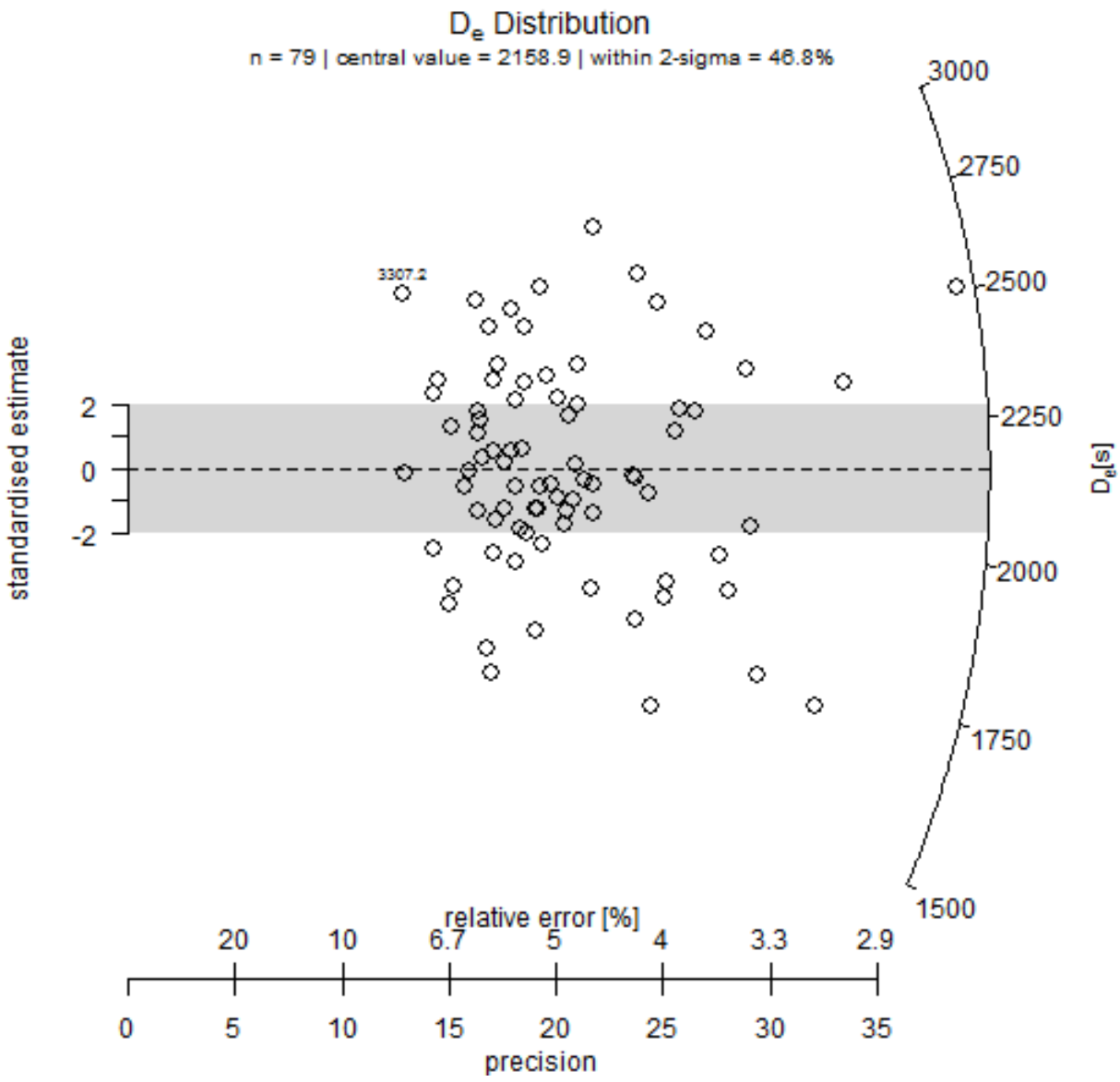
```
plot_DeDistribution(a, xlab = expression(paste(D[e], " [s]")))
```



### 3. Step: The radialplot

The function ***plot_RadialPlot*** will be used to plot a radial_plot. The needed arguments are the data set of ***a***, as well as ***zlab*** with the parameters De and [s] to label the z-axis, and finally ***zaxis.scale*** for the scaling of the z-axis.

```
plot_RadialPlot(a, zaxis.scale = seq(1500, 3000, by = 250), zlab =
expression(paste(D[e], "[s]")))
```



## 4. Step: The histogram

With the function ***plot_Histogram*** a histogram will be displayed. The requireded arguments are the data set of ***a***, as well as ***zlab*** with the parameters De and [s] to label the x-axis

```
plot_Histogram(a, xlab = expression(paste(D[e], "[s]")))
```