Workshop

R-package *Luminescence*

Einführung in die Plottfunktionen

Sebastian Kreutzer, Margret Fuchs, Michael Dietze, Manfred Fischer

Oktober 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

Inhaltsverzeichnis

Anmerkungen:

Bei den Erläuterungen zu den einzelnen Schritten werden

Funktionen rot,
Argumente grün und
Variablen blau hervorgehoben.

R-Befehle werden in einem farbig hinterlegten Rahmen dargestellt

```
dev.off()
```

Die Ausgabe nach Befehlsaufruf wird mit einem schwarzen Rahmen versehen.

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

1. Plotten von Kurven in eine Datei

1.Schritt

Sofern noch nicht geschehen wird als Erstes das Lumineszenz-Paket geladen.

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

2.Schritt

Mit Hilfe der Funktion ***readBIN2R*** wird ein Risoe-binfile geladen und der Variablen ***max*** zugewiesen. Als Argument wird nur der Pfad (grün) zur betreffenden Datei angegeben.

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

Beim Aufruf der Variablen ***max*** wird deren Inhalt ausgegeben.

```
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.Schritt

R stellt eine Vielzahl von Grafikformaten für den Export bereit. Exemplarisch sollen hier die Ausleucht- und TL-Kurven der in Schritt 2 eingelesenen Daten im PDF-Format ausgegeben werden. Dies geschieht mit Hilfe der Funktion ***pdf***, der drei Argumente übergeben werden. Das erste bestimmt den Dateinamen und gibt den Speicherort an. ***paper*** legt die Seitengröße und ***height*** die Höhe der einzelnen Diagramme fest.

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

4a.Schritt

Der Aufruf von ***par(mfrow = c(2, 1))*** unterteilt das Ausgabefenster in zwei Zeilen und eine Spalte, sodass insgesamt zwei Diagramme pro Seite geplottet werden können. Die Anzahl der Zeilen und Spalten werden mit dem Argument ***mfrow*** festgelegt. Der erste Wert legt die Anzahl der Zeilen, der zweite die Anzahl der Spalten fest.

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

siehe Plot 4a

4b.Schritt

In diesem Fall unterteilt der Aufruf von ***par(mfrow = c(3, 4))*** das Ausgabefenster in vier Zeilen und drei Spalten, sodass insgesamt zwölf Diagramme je Seite geplottet werden können.

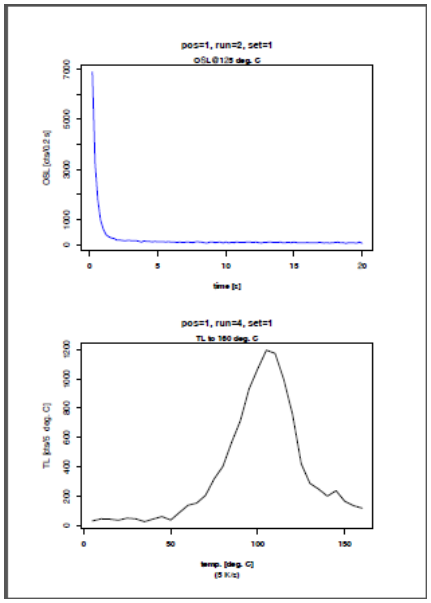
```
par(mfrow = c(3, 4))
```

siehe Plot 4b

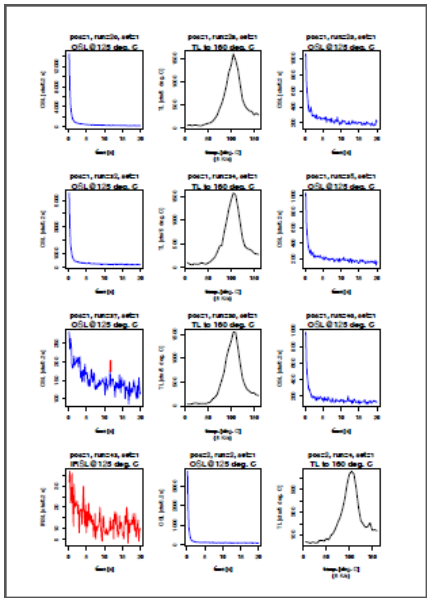
5.Schritt

Die Funktion ***plot_BINfileData*** plottet die einzelnen Diagramme.
Als Argument wird die Variable max, in der der Datensatz abgelegt ist übergeben.

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



Plot 4a



Plot 4b

6.Schritt

Mit ***dev.off()*** wird das R-eigene Grafikfenster geschlossen und das erzeugte PDF-Dokument kann geöffnet werden. Dieser Befehl ist zwingend auszuführen, da sonst nur ein leeres PDF-Dokument erstellt wird.

```
dev.off()
```

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

2. Das Plotten einer Ausleuchtkurve

1.Schritt

Hier wird der erste Datensatz [1] vom Datentyp "Liste" aus der Variablen ***max*** aufgerufen.

```
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.Schritt

Umwandlung des Datentyps und Variablenzuweisung.
Die Liste wird mit Hilfe der Funktion ***unlist*** in einen Vektor umgewandelt, sodass alle Listenelemente der Reihe nach auch im Vektor enthalten sind.Der neu erstellte Vektor wird der Variablen ***y*** zugewiesen.

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

3.Schritt

In diesem Schritt wird der Variablen ***zeit*** der erste Datensatz von ***max*** (max@METADATA) zugewiesen. Das Argument ***HIGH*** gibt das Zeitintervall der Messung an.

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

Beim Aufruf von ***zeit*** wird der Wert ausgegeben, hier = 20 (20 sec).

```
zeit
```

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

4.Schritt

Die Funktion ***length*** fragt die Länge (d.h. die Anzahl der Elemente) des Vektors ab. Gleichzeitig wird das Ergebnis der Abfrage der Variablen ***laenge*** zugewiesen.

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

Beim Aufruf von ***laenge*** wird die Anzahl der Elemente des Vektors ausgegeben, hier = 100.

```
laenge
```

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

5.Schritt

Die Funktion ***seq*** erstellt eine Zahlenfolge mit beliebigen Abständen. Das erste Argument gibt den Startwert an (***zeit/laenge***, d.h. 20/100 == 0.2), das zweite den Endwert und das letzte Argument ***by*** gibt den Abstand zwischen den einzelnen Werten an. Das Ergebnis des Funktionsaufrufs wird in der Variablen ***x*** abgelegt.

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

Beim Aufruf von ***x*** wird die Zahlenfolge (die Anzahl der Elemente) ausgegeben.

```
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.Schritt

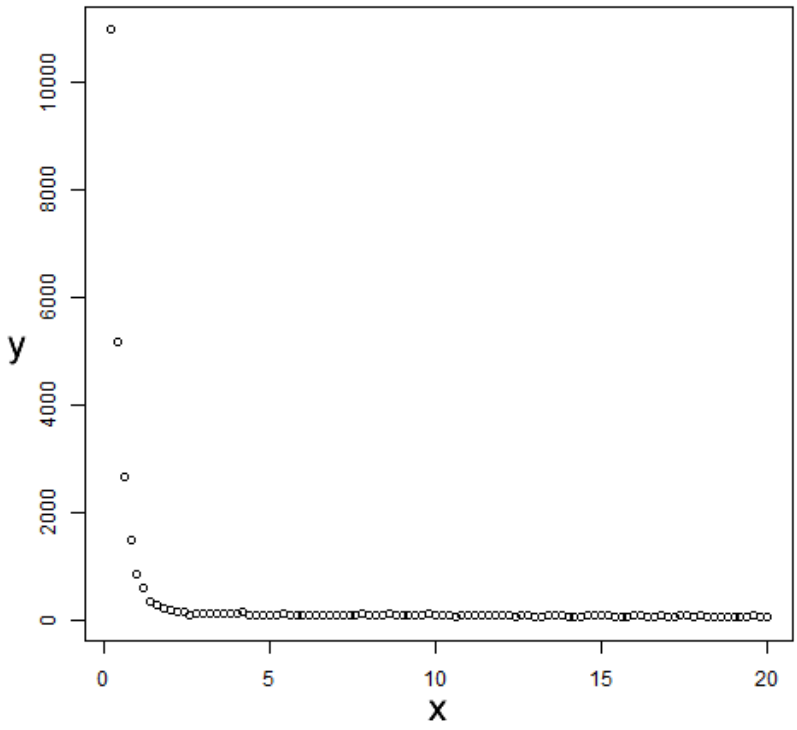
Die Variablen ***x*** und ***y*** werden in einen "data.frame" überführt und der variablen ***xy*** zugewiesen.

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

7.Schritt

Im nächsten Schritt wird ***xy*** mit Hilfe der R-eigenen Funktion ***plot*** dargestellt.

```
plot(xy)
```



3. Die Plotfunktion *plot_GrowthCurve*

1.Schritt

Die Funktion ***Analyse_SAR.OSLdata*** wird mit folgenden Argumenten aufgerufen:
max = Risoe.BINfileData
c(1:2) = das Integral von 1 bis 2 wird für die Auswertung herangezogen
c(85:100) = das Integral von 85 bis 100 wird als Untergrund abgezogen
Das Ergebnis wird der Variablen ***z*** zugewiesen.

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

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

2.Schritt

Aus der Variablen ***z*** wird der erste Datensatz extrahiert und der Variablen ***z1*** zugewiesen.

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

3.Schritt

Das Objekt ***z1*** wird mit der Funktion ***as*** in einen DataFrame umgewandelt. In R kann mit Hilfe der Funktion ***as*** ein beliebiger Datentyp in einen anderen "zwangsweise" umgewandelt werden.

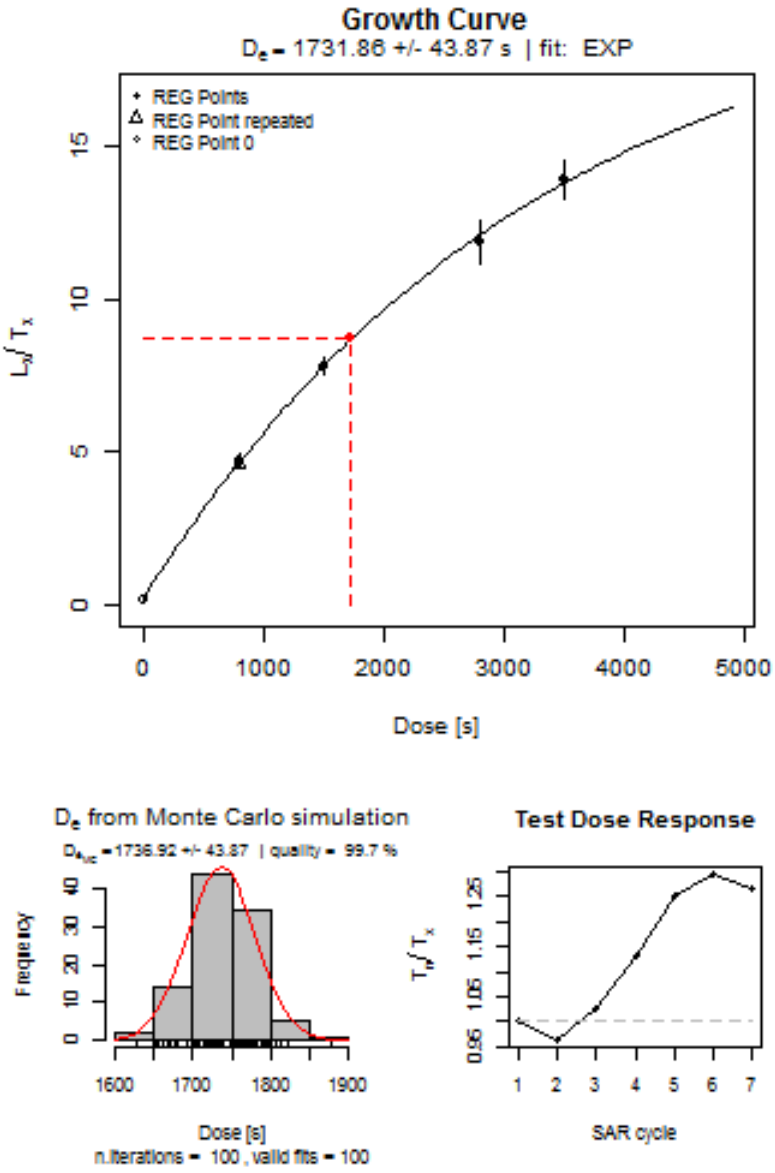
```
z.1 <- as.data.frame(z.1)
```

4.Schritt

Im letzten Schritt wir die Funktion ***plot_GrowthCurve*** aufgerufen.
Als Argument wird der Datensatz von ***z1*** übergeben.

```
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. Weitere Plotfunktionen:

4.1 plot_DeDistribution

4.2 plot_RadialPlot

4.3 plot_Histogram

1.Schritt: Einlesen der Daten

Im ersten Schritt wird mit der R-eigenen Funktion ***read.csv*** eine CSV_Datei mit zwei Spalten aufgerufen und das Ergebniss der Variablen ***a*** zugewiesen. Die Funktion wird mit drei Argumenten aufgerufen: Das erste gibt den Pfad zur CSV-Datei an, das zweite - header - die Beschriftung der Spalten, und das letzte - ***sep*** - legt als Seperator zwischen den einzelnen Daten ein Semikolon fest.

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

Ruft man ***a*** auf wird der Inhalt der Variablen dargestellt, und man kann die zwei Spalten ED und ED_Error erkennen.

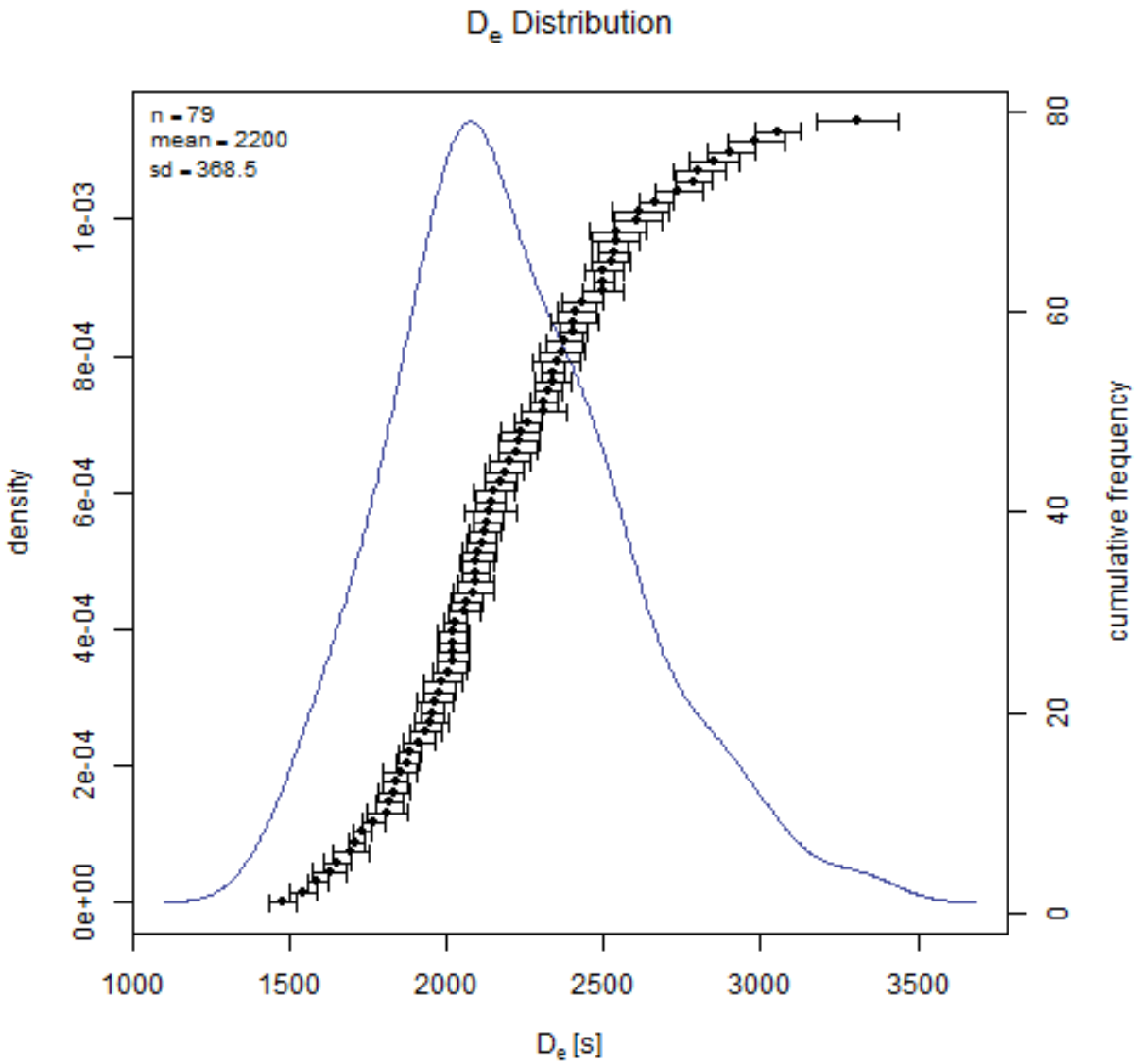
```
a
```

```
##      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
## 16   1950     106.24
## 17   2021     92.99
## etc.
```

2.Schritt: Plotten der De-Verteilung

Die De-Verteilung kann nun mit der Funktion ***plot_DeDistribution*** ausgedruckt werden. Als Argumente werden der Datensatz aus der Variablen ***a*** übergeben, sowie ***zlab*** mit den Parametern De und [s] zur x-Achsenbeschriftung.

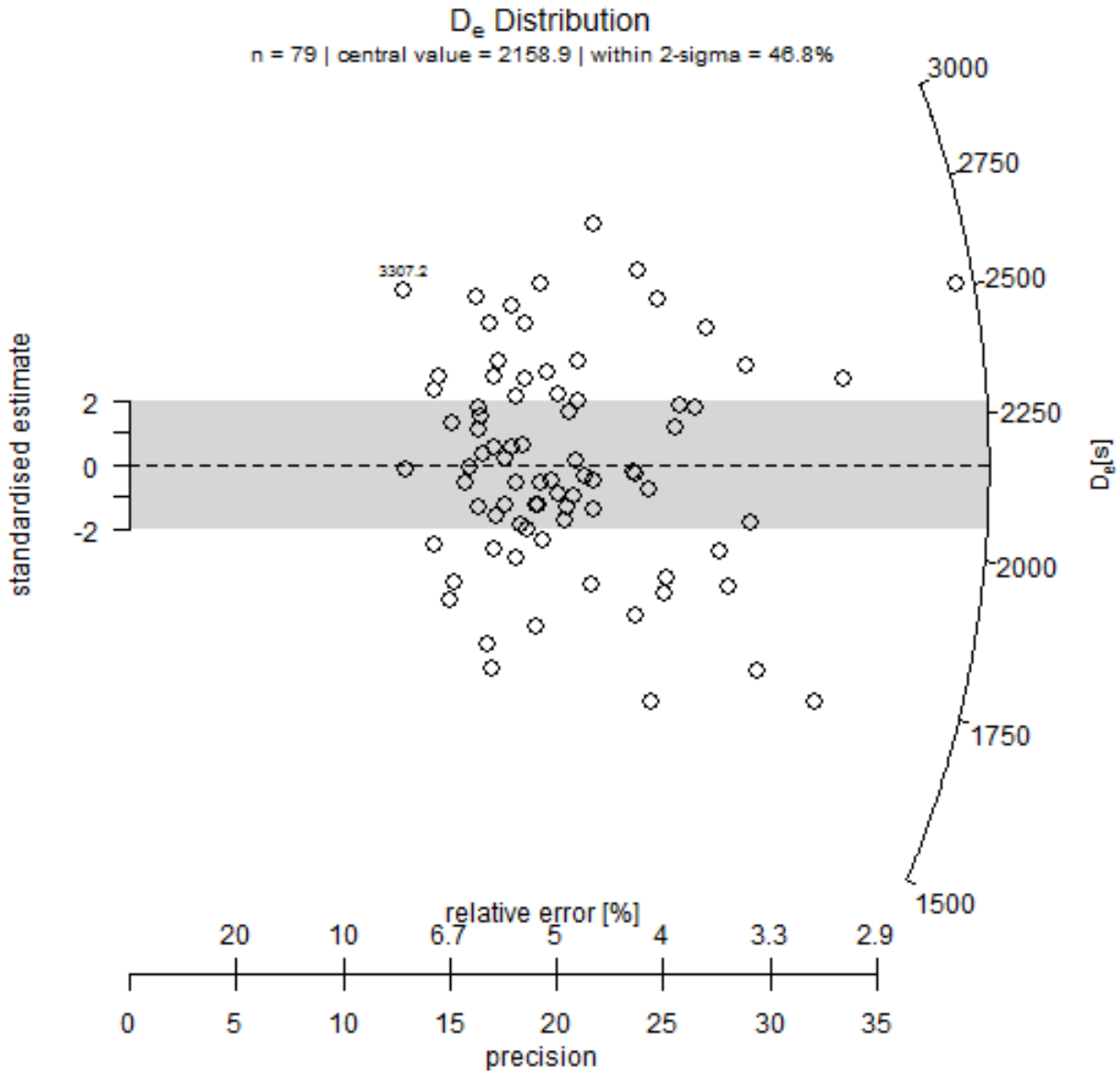
```
plot_DeDistribution(a, xlab = expression(paste(D[e], " [s]")))
```



3.Schritt: Der Radialplot

Der Radial-Plot wird mit der Funktion ***plot_RadialPlot*** ausgedruckt. Als Argumente werden der Datensatz aus der Variablen ***a*** übergeben, sowie ***zaxis.scale*** für die Skalierung der z-Achse, und ***zlab*** mit den Parametern De und [s] zur z-Achsenbeschriftung

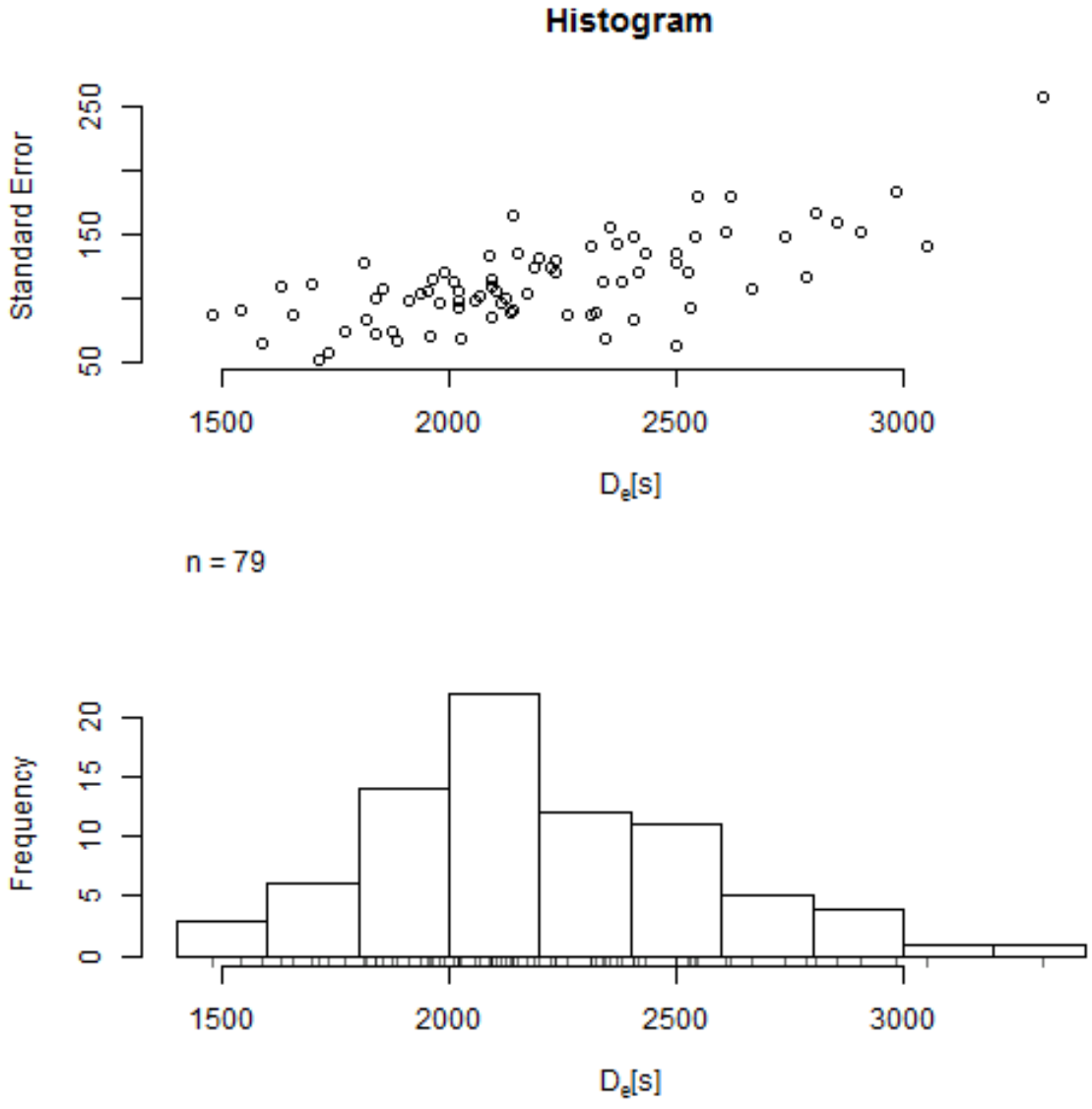
```
plot_RadialPlot(a, zaxis.scale = seq(1500, 3000, by = 250), zlab =
expression(paste(D[e], "[s]")))
```



4.Schritt: Das Histogramm

Mit der Funktion ***plot_Histogram*** wird ein Histogramm ausgedruckt. Als Argumente werden der Datensatz aus der Variablen ***a*** übergeben, sowie ***zlab*** mit den Parametern De und [s] zur x-Achsenbeschriftung.

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



5. Installationsanleitung von R mit R-Studio und Lumineszenzpacket

5.1 Windows

- Download des R-Packets
  http://cran.r-project.org/bin/windows/base/
  - Installation des R-Packets
- Download von R-Studio
  http://rstudio.org/download/desktop
  - Installation von R-Studio
- Einbinden des Lumineszenzpackets
  - Öffnen von R-Studio
  - Den Reiter *Packages* im unteren, rechten Fenster anklicken
  - *Install Packages* anklicken: Es öffnet sich ein Fenster, indem man im Auswahl-Feld *Install from Repository (Cran, Cranextra)* auswählt. Danach gibt man im Feld *Packages* den Begriff Luminescence ein und klickt auf *Install* →Fertig!

5.2 MacOsX

- Download des R-Packets
  http://cran.r-project.org/bin/macosx/
  - Installation des R-Packets
- Download von R-Studio
  http://rstudio.org/download/desktop
  - Installation von R-Studio
- Einbinden des Lumineszenzpackets
  - Öffnen von R-Studio
  - Den Reiter *Packages* im unteren, rechten Fenster anklicken
  - *Install Packages* anklicken: Es öffnet sich ein Fenster, indem man im Auswahl-Feld *Install from Repository (Cran, Cranextra)* auswählt. Danach gibt man im Feld *Packages* den Begriff Luminescence ein und klickt auf *Install* →Fertig!

5.3 Linux (Ubuntu)

- Download des R-Packets
  http://cran.r-project.org/bin/linux/
  - Installation des R-Packets
- Download von R-Studio
  http://rstudio.org/download/desktop
  - Installation von R-Studio
- Einbinden des Lumineszenzpackets
  - Öffnen von R-Studio
  - Den Reiter *Packages* im unteren, rechten Fenster anklicken
  - *Install Packages* anklicken: Es öffnet sich ein Fenster, indem man im Auswahl-Feld *Install from Repository (Cran, Cranextra)* auswählt. Danach gibt man im Feld *Packages* den Begriff Luminescence ein und klickt auf *Install* →Fertig!