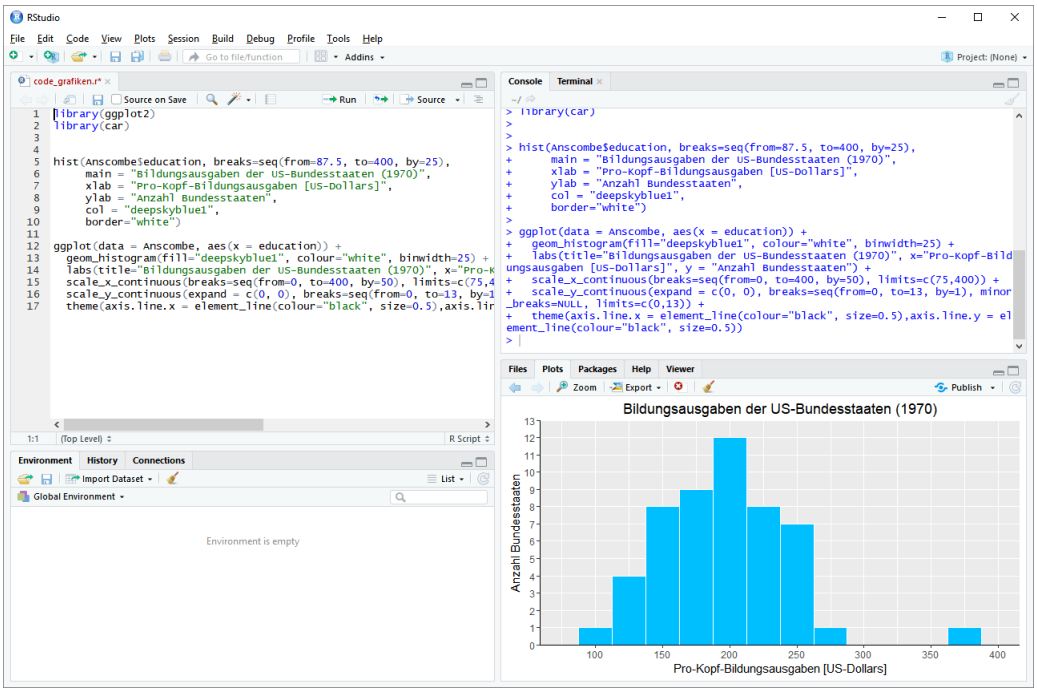
DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Campus Friedrichshafen

Messtechnik - Labor

Lehrveranstaltung Messtechnik für Wirtschaftsingenieure Fachrichtung E-Technik

Einführung in R



Name: .....

Gruppe: .....

Datum: .....

1. Aufgabe:

Installation und Kennenlernen von R und RStudio

a) Installieren Sie R von `www.freiwilligschlauwerden.de`.

R ist das Statistikprogramm, das Daten verarbeitet und die Ergebnisse ausgibt; es ist in der Rohfassung nicht sehr komfortabel zu bedienen. Deswegen arbeiten wir in diesem Kurs mit RStudio, einer sehr komfortablen und mächtigen integrierten Entwicklungsumgebung.

b) Installieren Sie RStudio ebenfalls von `www.freiwilligschlauwerden.de`.

c) Öffnen Sie RStudio. Klicken Sie in den linken unteren Bereich des Fensters ("Console"), tippen Sie

```
1 + 2
```

und schließen Sie die Eingabe mit Enter ab.

In der Kommandozeile der Konsole werden alle Anweisungen eingegeben und Textrückmeldungen des Programms ausgegeben; dazu gehören Ergebnisse, aber auch Hinweise, Warnungen und Fehlermeldungen, falls etwas nicht geklappt hat.

Die Kommandozeile eignet sich auch als Taschenrechner. Kennt man die Bedeutung einer Funktion nicht, kann man ein Fragezeichen voranstellen und bekommt eine Erklärung (rechts im Hilfebereich).

Bei Rechenoperationen gelten die Vorrangregeln der Mathematik (Potenz vor Punkt vor Strich). Der Dezimaltrenner ist ein Punkt (kein Komma). Exponential-, Logarithmus- bzw. Quadratwurzeln berechnet man über Funktionsaufrufe, das Argument steht in runden Klammern. Groß- und Kleinschreibung macht einen Unterschied. Stellt man einer Zeile ein `#` *-Zeichen voran, wird die Zeile von R nicht ausgeführt.*

Mit `help(<Befehl/Funktion>)` *bekommen Sie für die Argumente eine Hilfestellung im Plot-Fenster.*

d) Geben Sie folgende Ausdrücke ein und erklären Sie jeweils das Ergebnis:

```
2 + 3 * 4
(2 + 3) * 4
0.2 * 3 - 1.1
0,2 * 3
2^3^2
(2^3)^2
exp(1)
?exp
```

```
log(exp(1))
sqrt(16)
16^(1/2)
Sqrt(16)
# Das ist ein Kommentar.
```

Lösung:

```
2 + 3 * 4
```

```
## [1] 14
```

```
(2 + 3) * 4
```

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

```
0.2 * 3 - 1.1
```

```
## [1] -0.5
```

```
# 0,2 * 3 # Fehler, ',' wird nicht als Dezimalkomma
# akzeptiert
2^3^2
```

```
## [1] 512
```

```
(2^3)^2
```

```
## [1] 64
```

```
exp(1)
```

```
## [1] 2.718282
```

```
log(exp(1))
```

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

```
sqrt(16)
```

```
## [1] 4
```

```
16^(1/2)
```

```
## [1] 4
```

```
Sqrt(16)  # Fehler, sqrt() schreibt man mit kleinem 's'
```

```
## Error in eval(expr, envir, enclos): konnte Funktion "Sqrt" nicht finden
```

2. Aufgabe:

Variablen, Zuweisungen und Funktionen

Zahlen (und andere Objekte) können in R in Variablen gespeichert werden. Dazu kann der Zuweisungsoperator `=` *oder alternativ* `<-` *beispielsweise folgendermaßen verwendet werden:*

```
x <- 3.5
x2 <- 1.5  # funktioniert genauso mit x2 = 1.5
```

Mit diesen Variablen kann dann weitergerechnet werden. In Variablennamen dürfen Buchstaben, Ziffern (nicht als erstes Zeichen), Punkte und Unterstriche `(_)` *vorkommen. Diese Bezeichner dürfen keine Leerzeichen enthalten. Auch hier ist Groß- und Kleinschreibung zu beachten.*

a) Weisen Sie der Variablen x den Wert 4 zu. Weisen Sie dann der Variablen $x.2$ den folgenden Wert zu:

$$\sqrt{3x^2 + ln\left(\frac{1}{e^x}\right) + 5}$$

Funktionsaufrufe schreibt man in R mit einem Funktionsbezeichner, auf den direkt (keine Leerstelle!) ein Paar runder Klammern folgt. Innerhalb der runden Klammern können ein oder mehrere Argumente oder Parameter der Funktion stehen. Funktionen kann man auch verschachtelt aufrufen. Die Funktion `ls()` *gibt die in der aktuellen Sitzung definierten Objekte aus. Mit* `rm(`$< Var >$`)` *kann man eine Variable löschen, wenn man ihren Bezeichner anstatt* $< Var >$ *in die runden Klammern schreibt.*

b) Überlegen Sie was folgende Zeilen ausgeben und führen Sie diese dann in R aus, um Ihr Ergebnis zu überprüfen.

```
x
x.2
X
x + x.2
x.Produkt <- x * x.2
x.Produkt
x.Produkt <- x.Produkt * x
ls()
rm(x)
x
ls()
```

Außer Zahlen kann R auch mit Zeichenketten umgehen. Diese können in Objekten gespeichert werden, indem man die Zeichenkette in Anführungsstriche setzt. Zeichenketten, die Zahlen beinhalten werden nicht als Zahlen interpretiert. Man kann mit ihnen also nicht rechnen.

Tricks zur Ein- und Ausgabe:

- *Ist eine Eingabe in einer Zeile nicht vollständig, kann R das mit einem '+'-Zeichen anzeigen; die Eingabe kann dann vervollständigt werden.*
- *Sofortige Hilfe bei der Eingabe einer Funktion erhält man, wenn man nach Eingabe der ersten Buchstaben des Funktionsbezeichners die Tabulator-Taste betätigt. Die möglichen Funktionen werden dann zur Auswahl aufgelistet und können dann ausgewählt werden.*

- *Mit der Taste ↑ auf der Tastatur kann der letzte (oder bei zweimaligem Drücken der vorletzte usw.)*
- *Befehl wieder sichtbar gemacht und dann nochmals ausgeführt oder verändert werden.*
- *Im RStudio-Fenster finden Sie (meistens rechts oben) einen Reiter History. Auch dort werden alle eingegebenen Befehle abgespeichert.*
- *Im Reiter Environment werden alle Objekte der aktuellen Sitzung aufgelistet.*

c) Probieren Sie die angesprochenen Tricks zur Ein- und Ausgabe aus.

Lösung:

```
x <- 4
x.2 <- sqrt(3 * x^2 + log(1/exp(x)) + 5)
x.2
## [1] 7
X

## Error in eval(expr, envir, enclos): Objekt 'X' nicht gefunden
x + x.2
## [1] 11
x.Produkt <- x * x.2
x.Produkt
## [1] 28
x.Produkt <- x.Produkt * x
rm(x)
x  # Fehler: x gibt's ja nicht mehr, kann deswegen auch nicht ausgegeben werden

## Error in eval(expr, envir, enclos): Objekt 'x' nicht gefunden
```

3. Aufgabe:

Vektoren

Eine Urliste von Daten eines Merkmals wird in R durch einen Vektor repräsentiert. Zur Erzeugung eines Vektors dient die Funktion `c()`*. Die Einträge der Urliste werden dann zum Beispiel als Argumente von* `c()` *durch Kommata getrennt angegeben. Als Ausprägungen sind Zahlen oder Zeichenketten möglich. R versucht dann durch die Art der Argumente automatisch zu entscheiden, ob es sich um ein nominales oder ein metrisches Markmal handelt.*

a) Legen Sie eine Urliste für das Merkmal x an, das die Werte {1, 4, 2, 1.5} enthält. Geben Sie x aus.

Legen Sie ein weiteres Merkmal Geschlecht mit den Werten Mann, Frau, Frau, Frau an.

Geben Sie auch das Geschlecht aus. Das dritte Merkmal z soll die Werte {1, 2, 1, "1"} enthalten. Ist z für R nominal oder metrisch? Überprüfen Sie Ihre Entscheidung.

Vektoren aufeinanderfolgender ganzer Zahlen werden mit dem Doppelpunkt-Operator gebildet. $2:5$ *steht zum Beispiel für den Vektor mit den Zahlen 2, 3, 4, 5. Mit der Funktion* `seq()` *kann man genauer Vektoren als Folgen von Zahlen erzeugen.* `seq(from=2, to=3, by=0.2)` *erzeugt zum Beispiel den Vektor (2, 2.2, 2.4, 2.6, 2.8, 3). Mit* `rep()` *werden Werte oder ganze Vektoren vervielfacht als Vektor*

ausgegeben. Zum Beispiel ergibt `rep(c(1,2), 3)` *den Vektor (1,2,1,2,1,2). Die Hilfe-Seiten (Aufruf über* `?seq` *bzw.* `?rep`*) erklären die Details.*

b) Erzeugen Sie folgende Vektoren in R:

```
## [1] 5 6 7 8 9
##  [1] 10  9  8  7  6  5  4  3  2  1
## [1] -0.10 -0.08 -0.06 -0.04 -0.02  0.00
## [1] 10000 12500 15000 17500 20000
##  [1] -3 -2 -1  0  1  2 -3 -2 -1  0  1  2 -3 -2 -1  0  1
## [18]  2
##  [1]  5.0  6.0  7.0  8.0  9.0 10.0 10.1 10.2 10.3 10.4
## [11] 10.5
```

Rechenoperationen können zwischen (numerischen) Vektoren elementweise ausgeführt werden. Hat ein Vektor weniger Elemente als ein anderer, werden die Elemente vom Beginn des kürzeren Vektors einfach solange wiederholt, bis die Länge der beiden Vektoren gleich ist. Die Länge eines Vektors kann mir der Funktion `length()` *ausgelesen werden. Die Summe aller Elemente eines Vektors wird mit sum() errechnet. Beispielsweise ergibt mit* `x=1:5` *und* `y=c(10.1,10.5)` *die Summe* `x+y` *den Vektor (11.1, 12.5, 13.1, 14.5, 15.1). Analog funktioniert* `-, *, /.`

c) Gegeben sind die Vektoren

```
x <- 4:2
y <- seq(from = 0.1, to = 0.5, by = 0.1)
```

Erklären Sie, was folgende Ausdrücke ergeben und überprüfen Sie Ihr Ergebnis in R:

```
x + y
x * y
x^3 + 1
2 * x - 3 * y
n <- length(x + y)
sum(x + y)/n
```

d) Gegeben sind die Vektoren

```
x <- seq(from = 0, to = 100, by = 2)
y <- 100:1
```

Schreiben Sie die Ergebnisse folgender Ausdrücke auf und überprüfen Sie anschließend Ihr Ergebnis in R:

```
x[3]
y[c(1, 3, 10)]
x[1:4]
x[x > 91]
x[x > 20 & x <= 30]
y[y == 5 | y > 95 | y < 3]
```

Anmerkung: Die Ausgabe von Relationen wie $x < y$ auf Vektoren in R sind Vektoren mit den Ausprägungen `TRUE` *beziehungsweise* `FALSE`*. Diese sogenannten logischen Vektoren können zur Indizierung von Vektoren verwendet werden; Elemente mit einem Index von* `TRUE` *werden ausgewählt, die mit Wert* `FALSE` *werden übergangen.*

e) Was ergeben folgende Zeilen in R:

```
x <- seq(from = 0.2, to = 2, by = 0.3)
y <- -3:3
x < y
x^2 < x
Index <- x^2 < x
x[Index]
y[Index]
```

Lösung:

a)

```
x <- c(1, 4, 2, 1.5)  # Anlegen eines metrischen Merkmals x
# mit Ausprägungen für 4 Objekte
x  # Ausgabe
```

```
## [1] 1.0 4.0 2.0 1.5
```

```
Geschlecht <- c("Mann", "Frau", "Frau", "Frau")
Geschlecht  # nominales Merkmal, auch von 4 Objekten
```

```
## [1] "Mann" "Frau" "Frau" "Frau"
```

```
z <- c(1, 2, 1, "1")  # z ist für R nominal, da der letzte Wert
# als Zeichenkette eingegeben wurde
z
```

```
## [1] "1" "2" "1" "1"
```

b)

```
5:9
```

```
## [1] 5 6 7 8 9
```

```
10:1
```

```
##  [1] 10  9  8  7  6  5  4  3  2  1
```

```
seq(from = -0.1, to = 0, by = 0.02)
```

```
## [1] -0.10 -0.08 -0.06 -0.04 -0.02  0.00
```

```
seq(from = 10000, to = 20000, length.out = 5)
```

```
## [1] 10000 12500 15000 17500 20000
```

```
rep(-3:2, 3)
```

```
##  [1] -3 -2 -1  0  1  2 -3 -2 -1  0  1  2 -3 -2 -1  0  1
## [18]  2
```

```
c(5:10, seq(from = 10.1, by = 0.1, to = 10.5))
```

```
##  [1]  5.0  6.0  7.0  8.0  9.0 10.0 10.1 10.2 10.3 10.4
## [11] 10.5
```

c)

```
x <- 4:2
y <- seq(from = 0.1, to = 0.5, by = 0.1)
```

```
x + y
```

```
## [1] 4.1 3.2 2.3 4.4 3.5
```

```
x * y
```

```
## [1] 0.4 0.6 0.6 1.6 1.5
```

```
x^3 + 1
```

```
## [1] 65 28  9
```

```
2 * x - 3 * y
```

```
## [1] 7.7 5.4 3.1 6.8 4.5
```

```
n <- length(x + y)
sum(x + y)/n
```

```
## [1] 3.5
```

d)

```
x <- seq(from = 0, to = 100, by = 2)
y <- 100:1
```

```
x[3]
```

```
## [1] 4
```

```
y[c(1, 3, 10)]
```

```
## [1] 100  98  91
```

```
x[1:4]
```

```
## [1] 0 2 4 6
```

```
x[x > 91]
```

```
## [1]  92  94  96  98 100
```

```
x[x > 20 & x <= 30]
```

```
## [1] 22 24 26 28 30
```

```
y[y == 5 | y > 95 | y < 3]
```

```
## [1] 100  99  98  97  96   5   2   1
```

e)

```
x <- seq(from = 0.2, to = 2, by = 0.3)
y <- -3:3
x < y
```

```
## [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
```

```
x^2 < x
```

```
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
```

```
Index <- x^2 < x
x[Index]
```

```
## [1] 0.2 0.5 0.8
```

```
y[Index]
```

```
## [1] -3 -2 -1
```

4. Aufgabe:

Plotten

a) Plotten sie den Ausdruck $y=x^3+4x^2+5$ direkt mit dem Befehl `curve` (im Bereich -10 bis $+10$).

b) Erstellen Sie eine Funktion mit $y=x^3+4x^2+5$ und plotten diese mit dem Befehl `plot` (im Bereich -10 bis $+10$).

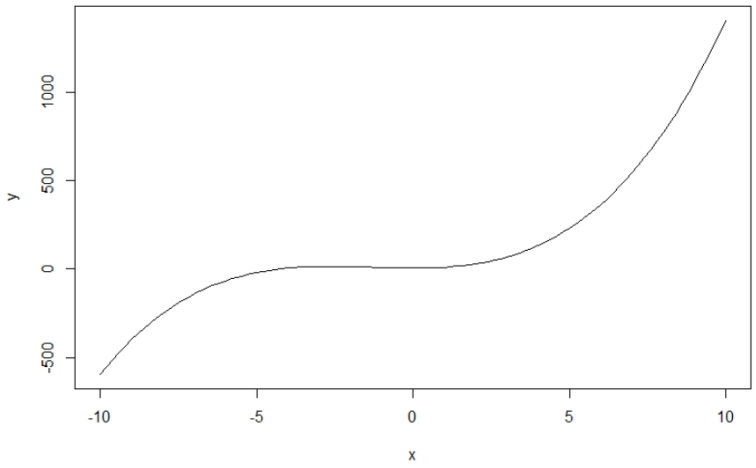
Lösung:

a) curve:

```
> curve(x^3+4*x^2+5,-10,10)
```

b) plot:

```
> y <- function(x) {
+   x^3+4*x^2+5
+ }
> plot(y,-10,10)
```



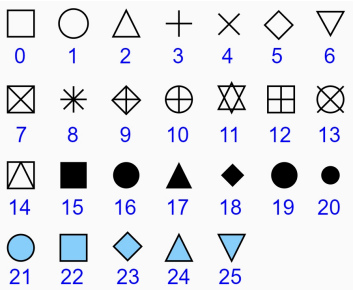
5. Aufgabe:

Fehlerbalken für Messwerte

a) Erstellen Sie einen Datensatz `Daten` mit der Funktion `data.frame` mit 5 Werten von 1 bis 5 (x) und den Werten für y (1.1, 1.5, 2.9, 3.8, 5.2), sowie der Standardabweichung (sd) (0.1, 0.2, 0.1, 0.3, 0.1).

b) Plotten Sie die y-Werte über x mit dem Befehl `plot` mit ausgefüllten Punkten und Achsenbeschriftung.

Default pcs Symbols:



c) Fügen Sie nun mit dem Befehl `arrows` die Standardabweichungen (sd.y) hinzu und plotten diese in das Diagramm in blau.

d) Erstellen Sie nun zusätzlich die Standardabweichungen für die x-Werte (sd.x) (0.1, 0.2, 0.2, 0.2, 0.1) und plotten diese ebenfalls (in rot) in das Diagramm.

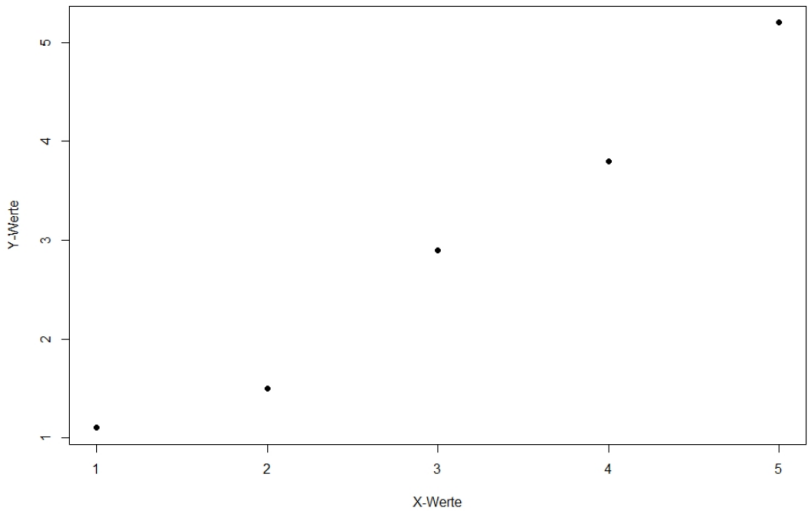
Lösung:

a) Datensatz:

```
# Datenpunkte (x, y) mit einer Standardabweichung (sd)
daten = data.frame(
  x  = c(1:5)
  , y  = c(1.1, 1.5, 2.9, 3.8, 5.2)
  , sd.y = c(0.1, 0.2, 0.1, 0.3, 0.1)
)
```

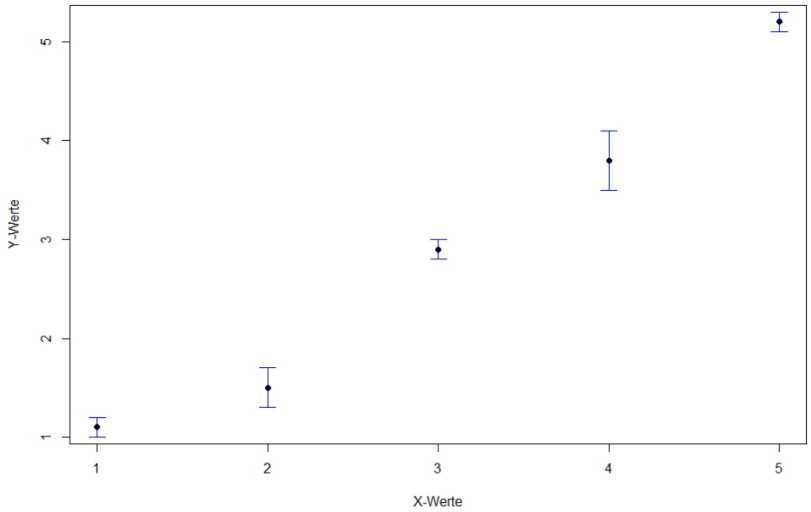
b) Daten Plot:

```
plot(daten$x, daten$y, type="p", pch=16, xlab="X-Werte", ylab="Y-Werte")
```



c) Plot Fehlerbalken y-Achse:

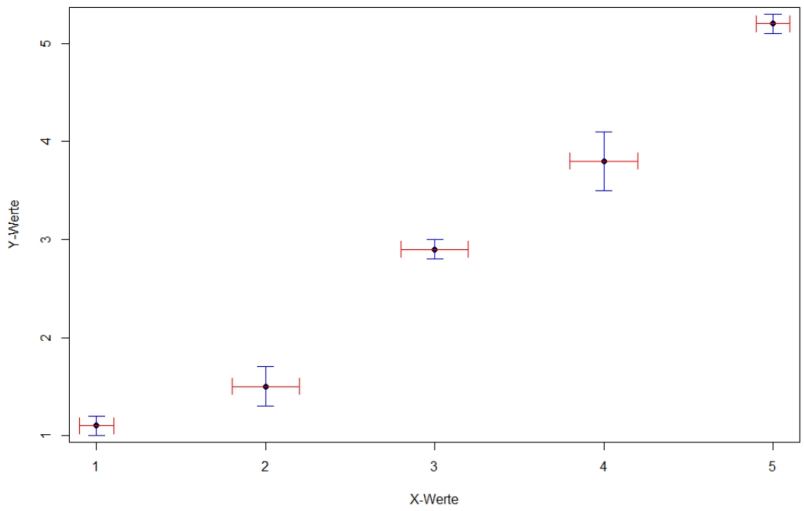
```
arrows(x0=daten$x, y0=daten$y-daten$sd.y, x1=daten$x, y1=daten$y+daten$sd.y, code=3, angle=90, length=0.1, col="blue", lwd=1)
```



d) Daten Standardabweichung x-Achse:

```
# Datenpunkte (x, y) mit einer Standardabweichung (sd)
daten = data.frame(
  x  = c(1:5)
  , y  = c(1.1, 1.5, 2.9, 3.8, 5.2)
  , sd.x = c(0.1, 0.2, 0.2, 0.2, 0.1)
  , sd.y = c(0.1, 0.2, 0.1, 0.3, 0.1)
)
```

e) Plot Fehlerbalken x-Achse:



6. Aufgabe:

Histogramm

a) Erstellen Sie einen Vektor x mit 1000 zufälligen, normalverteilten Zahlen (Mittelwert=5, Standardabweichung=1) mit dem Befehl `rnorm` und plotten Sie die Zahlen mit dem Befehl `hist`.

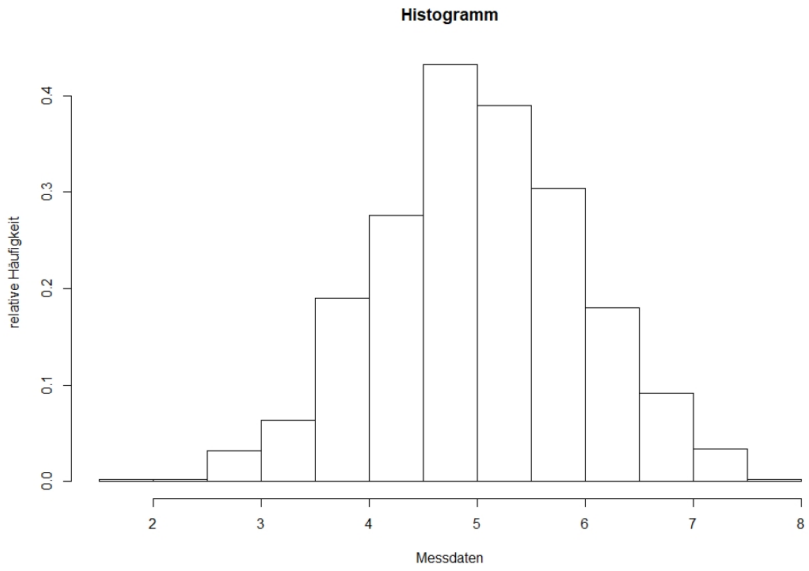
b) Plotten Sie in dieses Diagramm eine ideale Gaußkurve (Normalverteilung) mit denselben Werten für den Mittelwert und die Standardabweichung gestrichelt in rot ⇒ `curve`, `dnorm`.

c) Lassen Sie sich den Mittelwert, die Standardabweichung, den Median und die Min und Max-Werte ausgeben.

Lösung:

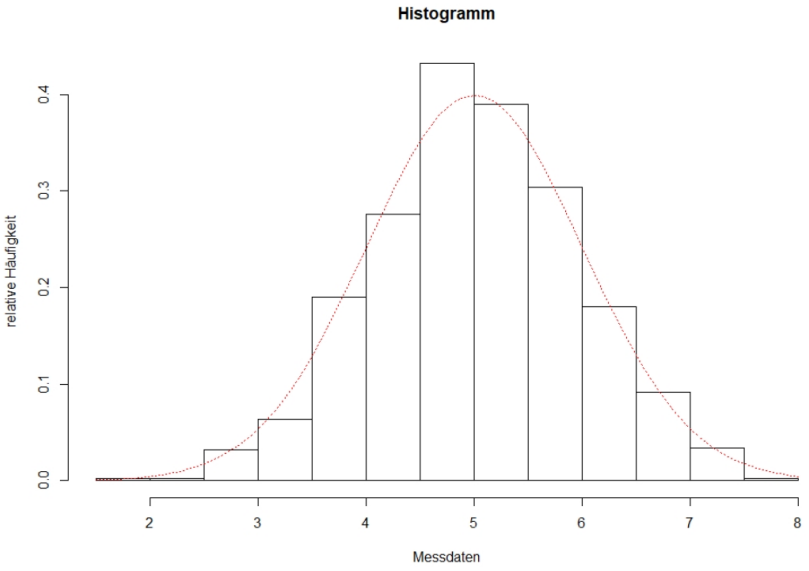
a) Histogramm:

```
x=rnorm(1000,5,1)
hist(x,freq=F,main="Histogramm",xlab="Messdaten",ylab="relative Häufigkeit")
```



b) Normalverteilung:

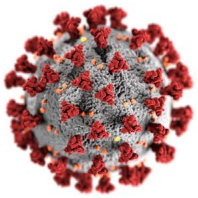
```
curve(dnorm(x,5,1), add=T, col="red", lty=3, lwd=1)
```



c) Zahlenwerte:

```
> mean(x)
[1] 5.014459
> sd(x)
[1] 0.9464302
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.853   4.422   5.002   5.014   5.643   7.622
```

Corona und das exponentielle Wachstum
Eine Analyse mit R



In diesem Abschnitt soll es um die Analyse der weltweiten Corona Fallzahlen und deren Vorhersage für die Zukunft gehen, falls keine eindämmende Maßnahmen getroffen werden.

Am 31. März 2020 wurden in Deutschland seit Ende Januar insgesamt ∼62'000 Covid19 Erkrankte gemeldet, in Italien ∼102'000 (Quelle: WHO: *https://covid19.who.int/table*).

Fragen die sich hieraus z.B. ergeben:

- Welches Wachstumsmodell liegt vor?
- Können basierend auf diesem Modell Vorhersagen gemacht werden?
- Wie unterschiedlich verlaufen die Fallzahlen in verschiedenen Ländern?
- Kann man aus den Daten z.B. schließen, dass der Verlauf in Italien grundsätzlich problematischer ist als in Deutschland?
- Wie verhält es sich mit den Todeszahlen?
- ...

7. Aufgabe (Datenaufbereitung):

a) Download der aktuellen Corona Falldaten von der WHO (*https://covid19.who.int/table*) Zahlen nur noch hier verfügbar:

b) ggf. Datensatz in UTF-8 umkodieren, Sonderzeichen entfernen... (mit Texteditor, z.B. Notepad^{++} für Windows)

c) Erstellen Sie ein neues Skript `"Covid.r"` und installieren Sie folgendes Package:

```
install.packages("data.table")
library(data.table)
```

d) Lesen Sie die Daten mit `read.csv` in R ein (Name: `COVID`) und wandeln diese in eine `data.table` Tabelle um.

e) Prüfen Sie die Datenbasis (insbesondere die Datumsspalte) mit: `str(COVID)`

f) Überführen Sie das Datumsformat der Datumsspalte in das Datumsformat von R

g) Prüfen Sie die Datenbasis erneut mit: `str(COVID)` und `head(COVID)`

h) Land auswählen und die Datensätze `BRD` und `ITA` erstellen (`Country_code=="DE", "IT"`)

i) Merkmal für die kummulierte Anzahl von Infinzierten auswählen und Datensätze `BRD_Infektionen` und `ITA_Infektionen` erstellen

j) Monat März auswählen und Datensatz `BRD_Infektionen_Maerz` und `ITA_Infektionen_Maerz` erstellen

k) Prüfen Sie die Datensätze wieder z.B. mit:

```
str(BRD_Infektionen_Maerz)

head(BRD_Infektionen_Maerz)

View(BRD_Infektionen_Maerz)
```

Lösung:

a-b) ohne Darstellung (Stand der Daten in dieser Musterlösung: **5. September 2020**)

c)
```
> install.packages("data.table")
Installing package into 'C:/Users/User/Documents/R/win-library/3.5'
(as 'lib' is unspecified)

  There is a binary version available but the source version is later:
           binary source needs_compilation
data.table 1.12.8 1.13.0              TRUE

  Binaries will be installed
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/data.table_1.12.8.zip'
Content type 'application/zip' length 2271847 bytes (2.2 MB)
downloaded 2.2 MB

package 'data.table' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\User\AppData\Local\Temp\RtmpcPPms7\downloaded_packages
> library(data.table)
data.table 1.12.8 using 6 threads (see ?getDTthreads).  Latest news: r-datatable.com
Warning message:
Paket 'data.table' wurde unter R Version 3.5.3 erstellt
>
```

d)
```
> options(scipen=999)  # statt e+10
> # Daten einlesen
> COVID = read.csv("COVID.csv",na="NA")
> COVID = data.table(COVID)
>
```

e)
```
> str(COVID)
Classes 'data.table' and 'data.frame':  39649 obs. of  8 variables:
 $ Date_reported    : Factor w/ 246 levels "2020-01-04","2020-01-05",..: 52 53 54 55 56 57 58 5
9 60 61 ...
 $ Country_code     : Factor w/ 215 levels " ","AD","AE",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ Country          : Factor w/ 216 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ WHO_region       : Factor w/ 7 levels "AFRO","AMRO",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ New_cases        : int  5 0 0 0 0 0 0 0 0 0 ...
 $ Cumulative_cases : int  5 5 5 5 5 5 5 5 5 5 ...
 $ New_deaths       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Cumulative_deaths: int  0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, ".internal.selfref")=<externalptr>
>
```

f)
```
> # Spalte Datum in Datumsformat überführen
> COVID$Date_reported=as.Date(COVID$Date_reported)
>
```

g)

```
> #Daten überprüfufen
> str(COVID)
Classes 'data.table' and 'data.frame':  39649 obs. of  8 variables:
 $ Date_reported    : Date, format: "2020-02-24" "2020-02-25" "2020-02-26" ...
 $ Country_code     : Factor w/ 215 levels " ","AD","AE",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ Country          : Factor w/ 216 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ WHO_region       : Factor w/ 7 levels "AFRO","AMRO",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ New_cases        : int  5 0 0 0 0 0 0 0 0 0 ...
 $ Cumulative_cases : int  5 5 5 5 5 5 5 5 5 5 ...
 $ New_deaths       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Cumulative_deaths: int  0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, ".internal.selfref")=<externalptr>
> head(COVID)
   Date_reported Country_code     Country WHO_region New_cases Cumulative_cases New_deaths
1:    2020-02-24           AF Afghanistan       EMRO         5                 5          0
2:    2020-02-25           AF Afghanistan       EMRO         0                 5          0
3:    2020-02-26           AF Afghanistan       EMRO         0                 5          0
4:    2020-02-27           AF Afghanistan       EMRO         0                 5          0
5:    2020-02-28           AF Afghanistan       EMRO         0                 5          0
6:    2020-02-29           AF Afghanistan       EMRO         0                 5          0
   Cumulative_deaths
1:                 0
2:                 0
3:                 0
4:                 0
5:                 0
6:                 0
>
```

h)

```
> # Land Auswählen
> BRD = COVID[Country_code=="DE"]
> ITA = COVID[Country_code=="IT"]
>
```

i)

```
> # Merkmal auswählen
> BRD_Infektionen = BRD[,.(Date_reported,Cumulative_cases)]
> ITA_Infektionen = ITA[,.(Date_reported,Cumulative_cases)]
>
```

j)

```
> # Monat Maerz auswaehlen
> BRD_Infektionen_Maerz= subset(BRD_Infektionen, Date_reported>as.Date('2020-03-01') & Date_reported<as.Date('2020-04-01'))
> ITA_Infektionen_Maerz=subset(ITA_Infektionen, Date_reported>as.Date('2020-03-01') & Date_reported<as.Date('2020-04-01'))
>
```

k)

```
> #Daten überprüfufen
> str(BRD_Infektionen_Maerz)
Classes 'data.table' and 'data.frame':  31 obs. of  2 variables:
 $ Date_reported   : Date, format: "2020-03-01" "2020-03-02" "2020-03-03" ...
 $ Cumulative_cases: int  111 129 157 196 262 400 684 847 902 1139 ...
 - attr(*, ".internal.selfref")=<externalptr>
> head(BRD_Infektionen_Maerz)
   Date_reported Cumulative_cases
1:    2020-03-01              111
2:    2020-03-02              129
3:    2020-03-03              157
4:    2020-03-04              196
5:    2020-03-05              262
6:    2020-03-06              400
>
```

```
> View(BRD_Infektionen_Maerz)
>
```

Filter

	Date_reported	Cumulative_cases
1	2020-03-01	111
2	2020-03-02	129
3	2020-03-03	157
4	2020-03-04	196
5	2020-03-05	262
6	2020-03-06	400
7	2020-03-07	684
8	2020-03-08	847
9	2020-03-09	902
10	2020-03-10	1139
11	2020-03-11	1296
12	2020-03-12	1567
13	2020-03-13	2369
14	2020-03-14	3062
15	2020-03-15	3795
16	2020-03-16	4838
17	2020-03-17	6012
18	2020-03-18	7156
19	2020-03-19	8198
20	2020-03-20	14138
21	2020-03-21	18187
22	2020-03-22	21463
23	2020-03-23	24774
24	2020-03-24	29212
25	2020-03-25	31554
26	2020-03-26	36508
27	2020-03-27	42288
28	2020-03-28	48582
29	2020-03-29	52547
30	2020-03-30	57298
31	2020-03-31	61913

Showing 1 to 31 of 31 entries

8. Aufgabe (Datenanalyse):

a) Erstellen Sie für Deutschland im Maerz (`BRD_Infektionen_Maerz`) und `ITA_Infektionen_Maerz`)) ein einfaches Punktdiagramm mit dem Befehl: `plot`.

Welcher allgemeinen Funktion folgen die Fallzahlen rein visuell gesehen am ehesten, vor allem in der ersten Märzhälfte?

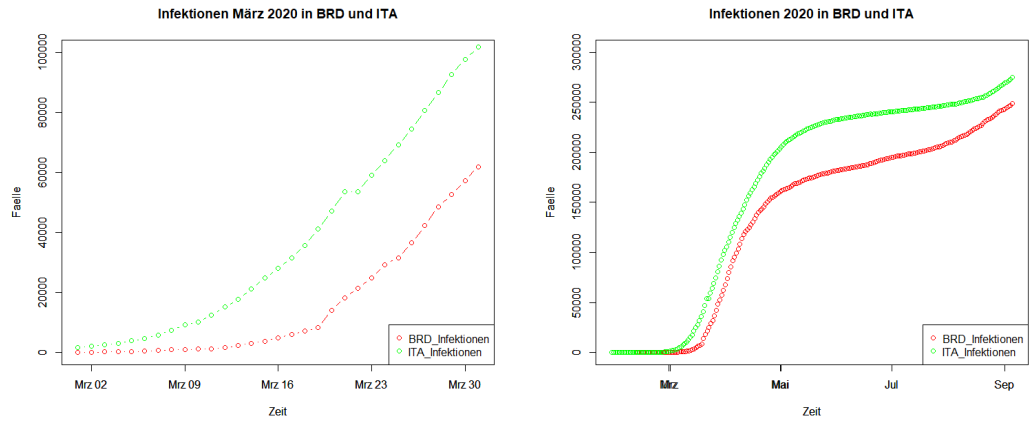
Was kann man bezüglich den Infinziertenzahlen nach der TV-Ansprache von Angela Merkel mit Lockdown am 19.3.2020 erkennen?

Erstellen Sie die Diagramme ebenfalls für `BRD_Infektionen` und `ITA_Infektionen)`.

b) Wie korrelieren die beiden Datensätze im März und insgesamt?
Stellen Sie dazu die Infiziertenzahlen von Deutschland und Italien gegenüber und bestimmen Sie die Korrelationskoeffizienten nach Pearson mit dem Befehl: `cor.test`.

Lösung:

a)
```
> # Plots
> plot(BRD_Infektionen_Maerz,ylim=c(0,100000),main="Infektionen März 2020 in BRD und ITA",type="b",ylab="Faelle",xlab="Zeit",col="red")
> par(new=TRUE)
> plot(ITA_Infektionen_Maerz,ylim=c(0,100000),type="b",ylab="Faelle",xlab="Zeit",col="green")
> legend("bottomright",c("BRD_Infektionen","ITA_Infektionen"),col=c("red","green"),pch=1)
>
> plot(BRD_Infektionen,ylim=c(0,300000),main="Infektionen 2020 in BRD und ITA",ylab="Faelle",xlab="Zeit",col="red")
> par(new=TRUE)
> plot(ITA_Infektionen,ylim=c(0,300000),ylab="Faelle",xlab="Zeit",col="green")
> legend("bottomright",c("BRD_Infektionen","ITA_Infektionen"),col=c("red","green"),pch=1)
>
```

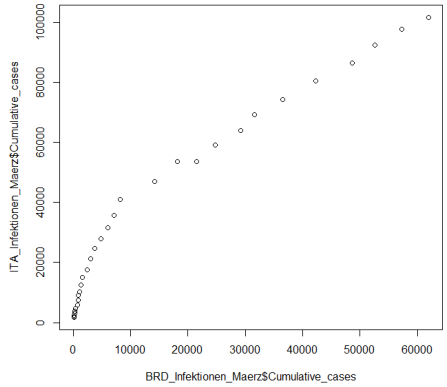


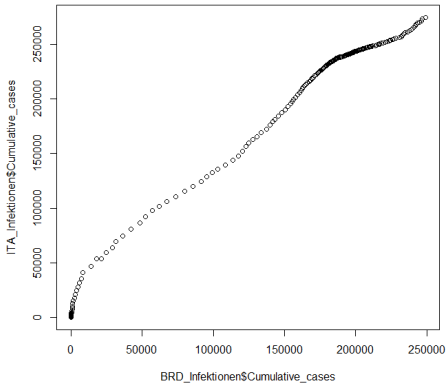
Das Modell für den März enstspricht am ehesten einer Exponentialverteilung (am Beginn der Epidemie).

Ab dem Lockdown Mitte/Ende März in der BRD steigen die Infiziertenzahlen nicht mehr exponentiell an, sondern schwächen sich ab.

b)

```
> # Korrelationstest Maerz
> cor.test(BRD_Infektionen_Maerz$Cumulative_cases,ITA_Infektionen_Maerz$Cumulative_cases)

        Pearson's product-moment correlation

data:  BRD_Infektionen_Maerz$Cumulative_cases and ITA_Infektionen_Maerz$Cumulative_cases
t = 21.195, df = 29, p-value < 0.00000000000000022
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9364762 0.9851994
sample estimates:
      cor
0.9692042

> plot(BRD_Infektionen_Maerz$Cumulative_cases, ITA_Infektionen_Maerz$Cumulative_cases)
>
> # Korrelationstest gesamt
> # Anpassung Zeilen auf 221 Zeilen
> BRD_Infektionen = BRD_Infektionen[-1,]  #ITA 221, BRD 222
> cor.test(BRD_Infektionen$Cumulative_cases,ITA_Infektionen$Cumulative_cases)

        Pearson's product-moment correlation

data:  BRD_Infektionen$Cumulative_cases and ITA_Infektionen$Cumulative_cases
t = 123.91, df = 219, p-value < 0.00000000000000022
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9908071 0.9945841
sample estimates:
      cor
0.9929431

> plot(BRD_Infektionen$Cumulative_cases, ITA_Infektionen$Cumulative_cases)
>
```





9. Aufgabe (Modellbildung):

Um das genaue Modelle zu erhalten, stellen wir ein Exponential-Modell (decay-model) nach folgender Gleichung auf:

$$N(t) = N_0 a^t \qquad \text{oder gleichwertig}$$

$$N(t) = N_0 e^{\lambda t} \qquad \text{mit } \lambda = \ln(a)$$

Einfaches Modell mit Schätzung von a (Wahrscheinlichkeitsrate) bzw. λ (Wahrscheinlichkeitskonstante) mit zwei Werten:

a) Bestimmen Sie allgemein den Parameter a und λ.

b) Bestimmen Sie den Wert für a, indem Sie zwei Werte aus dem Datensatz (`BRD_Infektionen_Maerz`) einsetzen. Nehmen Sie den Startzeitpunkt 2020-03-01 (Tag 0) an und den zweiten Wert am Tag 2020-03-19 (Tag 18).

c) Erstellen Sie ein Schaubild mit dem Modell für die BRD im Monat März und vergleichen Sie dies mit den Originaldaten im selben Plotfenster.

d) Erstellen Sie abschließend noch ein Blockdiagramm mit: `barplot`.

Lösung:

a) Wir können die Basis a bzw. λ durch Logarithmieren bestimmen:

$$N(t) = N_0 a^t \qquad N(t) = N_0 e^{\lambda \cdot t} \quad \text{mit} \quad \lambda = \ln(a)$$

$$\frac{N(t)}{N_0} = a^t \qquad \frac{N(t)}{N_0} = e^{\lambda \cdot t}$$

$$\log\left(\frac{N(t)}{N_0}\right) = \log(a^t) \qquad \ln\left(\frac{N(t)}{N_0}\right) = \ln\left(e^{\lambda \cdot t}\right)$$

$$\log\left(\frac{N(t)}{N_0}\right) = t \cdot \log(a) \qquad \ln\left(\frac{N(t)}{N_0}\right) = \lambda \cdot t$$

$$\frac{\log\left(\frac{N(t)}{N_0}\right)}{t} = \log(a) \qquad \lambda = \frac{\ln\left(\frac{N(t)}{N_0}\right)}{t}$$

$$\Rightarrow \quad a = 10^{\left(\frac{\log\left(\frac{N(t)}{N_0}\right)}{t}\right)} \qquad \Rightarrow \quad a = e^{\left(\frac{\ln\left(\frac{N(t)}{N_0}\right)}{t}\right)}$$

b) Wir setzen für `BRD_Infektionen_Maerz` → nachschauen im Modell `View(...)`:

$N_0 = N(\text{2020-03-01}) = \texttt{BRD\_Infektionen\_Maerz}[1] = 111$
$N_{18} = N(\text{2020-03-19}) = \texttt{BRD\_Infektionen\_Maerz}[19] = 8198$

und erhalten folgendes Modell:

$N(t) = N_0 a^t \quad$ mit

$$a = 10^{\left(\frac{\log\left(\frac{N_{18}}{N_0}\right)}{18}\right)} = 10^{\left(\frac{\log\left(\frac{8198}{111}\right)}{18}\right)} \approx 10^{\left(\frac{\log(73.856)}{18}\right)} = 10^{0.1038} = 1.270$$

$\Rightarrow \quad N(t) = 111 \cdot (1.270)^t \quad \rightarrow$ Wachstumsrate in Prozent $\widehat{=} 1 - a = 27\%$ pro Tag

bzw. mit $\lambda = \ln(a) = \ln(1.270) = 0,239$

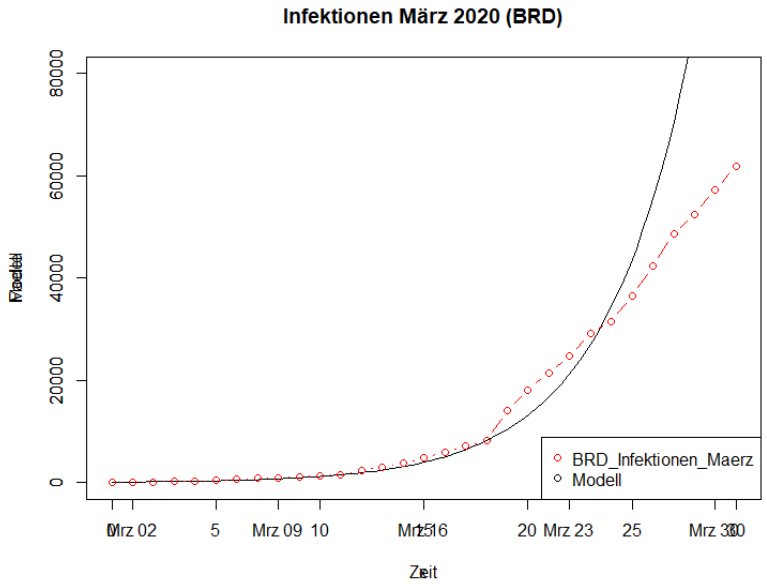
$\Rightarrow \quad N(t) = 111 \cdot e^{0,239 \cdot t}$

In R:

```
> a0=BRD_Infektionen_Maerz$Cumulative_cases[1]   #1:  111
> a0
[1] 111
> a1=BRD_Infektionen_Maerz$Cumulative_cases[19]  #19: 8198
> a1
[1] 8198
> a2=a1/a0
> a2
[1] 73.85586
> a3=log10(a2)
> a3
[1] 1.868385
> a4=a3/18
> a4
[1] 0.1037992
> a=10^a4
> a
[1] 1.269987
> Modell=function(x){a0*(a^x)}
> Modell(0)
[1] 111
> Modell(18)
[1] 8198
>
```

c)

```
> plot.new()
> plot(Modell,0,30,ylim=c(0,80000),axisnames=FALSE,axis=FALSE)
There were 12 warnings (use warnings() to see them)
> par(new=TRUE)
There were 24 warnings (use warnings() to see them)
> plot(BRD_Infektionen_Maerz,ylim=c(0,80000),main="Infektionen März 2020 (BRD)",type="b",ylab="Faelle",xlab="Zeit",col="red",)
> legend("bottomright",c("BRD_Infektionen_Maerz","Modell"),col=c("red","black"),pch=1)
There were 24 warnings (use warnings() to see them)
>
```



d)

```
> barplot(BRD_Infektionen_Maerz$Cumulative_cases, ylim =c(0,80000),main="Corona Infektionen Maerz 2020 (BRD)",type="b",ylab="Faelle",xlab="Tage",col="red")
```