R Kurs Übungen

1. Übungen zum Erstellen, Indizieren und Berechnungen mit Vektoren in R

Aufgabe 1:

Erstellen Sie einen Vektor, der die Zahlen 3, 9, 7, 6, 4, 1 enthält.

Berechnen Sie anschließend den Mittelwert der Elemente im Vektor und die Quadratwurzel der einzelnen Elemente.

Lösung:

```
# Erstellen des Vektors
> x <- c(3, 9, 7, 6, 4, 1)

# Ausgabe der Elemente des Vektors
> x
[1] 3 9 7 6 4 1

# Berechnen vom Mittelwert aller Vektorelemente
> mean(x)
[1] 5

# Berechnen der Quadratwurzel der einzelnen Elemente
> sqrt(x)
[1] 1.732051 3.000000 2.645751 2.449490 2.000000 1.000000
```

Aufgabe 2:

Erstellen Sie einen Vektor mit den ganzen Zahlen 1, 10, 100 als Datentyp integer. Schlagen Sie zwei Vorgehensweisen vor. Überprüfen Sie den Datentyp durch eine Abfrage.

Erstellen des Vektors erste Möglichkeit

> y <- c(10L, 100L, 1000L)

Ausgabe der Elemente des Vektors

> y

[1]   10 100 1000

Abfrage des Dateityps

> class(y)

[1] "integer"

Erstellen des Vektors zweite Möglichkeit

> y2 <- as.integer(c(10, 100, 1000))

> class(y2)

[1] "integer"

Aufgabe 3:

Erstellen Sie einen Vektor mit den folgenden Elementen: "Arbeiten" "mit" "R" "ist" "einfach". Welcher Datentyp liegt vor?

Erstellen des Vektors

> ab <- c("Arbeiten", "mit", "R", "ist", "einfach")

> ab

[1] "Arbeiten" "mit"   "R"   "ist"   "einfach"

```
# Abfrage des Dateityps
> class(ab)
[1] "character"
```

Aufgabe 4:

Erstellen Sie einen Vektor, der jeweils nur die ungeraden Zahlen von 1 bis 100 enthält. Versuchen Sie, die kürzeste Schreibweise zu finden.

```
#  Erstellen einer Sequenz von Zahlen als Vektor
> z <-seq(from = 1, to = 100, by = 2)

> z
 [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
[26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

Aufgabe 5:

Erstellen Sie einen Vektor x, der die Zahlen von 1 bis 5 enthält und einen Vektor y, der die Zahlen von 7 bis 20 enthält. Kombinieren Sie im Anschluss beide Vektoren zu einem Vektor z. Löschen Sie danach die Vektoren x und y aus dem Speicher. Sortieren Sie danach die Elemente von z in absteigender Reihenfolge.

```
#  Erstellen der Vektoren
> x <- c(1:5)
> y <- c(7:20)

# Zusammenfügen der Vektoren
> z <- c(x, y)
> z
 [1]  1  2  3  4  5  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
#  Löschen der Vektoren
> rm(x)
> rm(y)
> x
Fehler: Objekt 'x' nicht gefunden
> y
Fehler: Objekt 'y' nicht gefunden

# Sortieren des Vektors mit sort() under der Angabe decreasing = TRUE
> z2 <- sort(z, decreasing = TRUE)
> z2
 [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  5  4  3  2  1
```

Aufgabe 6:

Berechnen Sie den Mittelwert des 2., 7., und 9. Elements des Vektors z aus Aufgabe 5. Indizieren Sie dazu zunächst die entsprechenden Elemente.

```
# Lösung in zwei Schritten
# Indizieren des Vektors
> w <- z[c(2, 7, 9)]
> w
[1]  2  8 10

# Bilden des Mittelwerts
> mean(w)
[1] 6.666667

# Lösung in einem Schritt
> mean(z[c(2, 7, 9)])
[1] 6.666667
```

2. Übungen zu Matrizen

Aufgabe 1a:

Erstellen Sie eine Martrix mit den Zahlen von 1 bis 10 die folgendermaßen aussieht:

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

Erstellen der Matrix

```
> M <- matrix(c(1:10), nrow = 5)
```

Aufgabe 1b:

Im Folgenden sollen die Zahlen statt spaltenweise zeilenweise eingetragen werden.

Erstellen der neuen Matrix mit Hilfe des Arguments byrow

```
> M2 <- matrix(c(1:10), nrow = 5, byrow = TRUE)
> M2
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
[5,]    9   10
```

Aufgabe 1c:

Lesen Sie aus dieser Matrix das Element in der 2. Zeile und der 3. Spalte, also die Zahl "6" aus.

```
# Das Indizieren der Matrix erfolgt durch eckige Klammern

> M2[3,2]

[1] 6
```

Aufgabe 1d:

Subtrahieren Sie von jedem Element der 2. Spalte die Zahl 2. Bilden Sie anschließend den Mittelwert des neuberechneten Vektors.

```
# Im Vektor x stehen nun alle Elemente der Spalte 2 minus 2

> x <- c(M2[,2]-2)

> x

[1] 0 2 4 6 8

# Nun wird noch der Mittelwert gebildet

> mean(c(M2[,2]-2))

[1] 4
```

Aufgabe 1e:

Erstellen Sie einen Vektor M3 aus dem Vektor M2, in dem die Zahlen der Spalte 2 durch die Zahlen der Spalte 2 minus 2 ersetzt sind.

```
# Die neue Matrix wird aus den Elementen von M2 erstellt
> M3 <- cbind(M2[,1], (M2[,2]-2))

> M3
     [,1] [,2]
[1,]    1    0
[2,]    3    2
[3,]    5    4
[4,]    7    6
[5,]    9    8
```

Aufgabe 1f:

Fügen Sie dem neugebildeten Vektor M3 noch eine Zeile 6 mit den Elementen 12 und 14 hinzu. Danach fügen Sie der Matrix noch eine 3. Spalte mit den Zahlen 20 bis 25 hinzu.

```
# Die neue Zeile wird mit rbind() hinzugefügt.
> M3 <- rbind(M3, c(12, 14))
> M3
     [,1] [,2]
[1,]    1    0
[2,]    3    2
[3,]    5    4
[4,]    7    6
[5,]    9    8
[6,]   12   14
```

Die neue Spalte wird mit rbind() hinzugefügt.

```
> M3<- cbind(M3, c(20:25))
> M3
     [,1] [,2] [,3] [,4]
[1,]    1    0   20   20
[2,]    3    2   21   21
[3,]    5    4   22   22
[4,]    7    6   23   23
[5,]    9    8   24   24
[6,]   12   14   25   25
```

Aufgabe 2:

Erstellen Sie eine Matrix mit zwei Zeilen und zwei Spalten mit den Zahlen 1 bis 4, reihenweise eingetragen. Die Zeilen sollen die Namen "a1" und "a2" tragen, die Spalten die Namen "b1" und "b2".

Zunächst definieren wir die Namen der Zeilen und Spalten in Vektoren

```
> colnames <- c("b1", "b2")
> rownames <- c("a1", "a2")
```

In der neu erstellten Matrix werden die Namen als "dimnames" hinzugefügt

```
> M4 <- matrix(c(1:4), ncol = 2, byrow = TRUE, dimnames = list(colnames, rownames))

> M4
   a1 a2
b1  1  2
b2  3  4
```

Aufgabe 3:

Erstellen Sie folgenden Vektor und versuchen Sie dabei, die kürzeste Schreibweise zu finden:

```
      [,1] [,2] [,3] [,4] [,5]
 [1,]  100   80   60   40   20
 [2,]   98   78   58   38   18
 [3,]   96   76   56   36   16
 [4,]   94   74   54   34   14
 [5,]   92   72   52   32   12
 [6,]   90   70   50   30   10
 [7,]   88   68   48   28    8
 [8,]   86   66   46   26    6
 [9,]   84   64   44   24    4
[10,]   82   62   42   22    2
```

```
#  So geht es schnell
> M <- matrix(seq(from = 100, to = 1, by = -2), nrow = 10)
```

3. Übungen zu Dataframes

Aufgabe 1a:

Erstellen Sie folgendes Dataframe als "data":

	Pflanzen_Nr	fertile_Samen	abortierte_Samen
1	At1	377	10
2	At2	879	30
3	At3	216	41
4	At4	93	71
5	At5	98	22
6	At6	103	6

Zum Erstellen des Dataframes schreiben wir

```
> data <- data.frame(Pflanzen_Nr = c("At1", "At2", "At3", "At4", "At5", "At6"), fertile_Samen = c(377, 879, 216, 93, 98, 103), abortierte_Samen = c(10, 30, 41, 71, 22, 6), stringsAsFactors = FALSE)
```

Aufgabe 1b:

Lassen Sie sich die statistische Zusammenfassung für das Dataframe "data" wiedergeben.

```
> summary(data)
 Pflanzen_Nr        fertile_Samen    abortierte_Samen
 Length:6           Min.   : 93.00   Min.   : 6.00
 Class :character   1st Qu.: 99.25   1st Qu.:13.00
 Mode  :character   Median :159.50   Median :26.00
                    Mean   :294.33   Mean   :30.00
                    3rd Qu.:336.75   3rd Qu.:38.25
                    Max.   :879.00   Max.   :71.00
```

Aufgabe 1c:

Leider wurden in dem Dataframe die Angaben vergessen, wie viele Schoten insgesamt angesehen wurden. Fügen Sie die Angaben als Spalte "ausgezaehlte_Schoten" hinzu.

Es handelt sich von At1 bis At6 um 6, 18, 4, 3, 2 und 2 Schoten jeweils.

Zunächst erstellen wir die neue Spalte des Dataframes

```
> data$ausgezaehlte_Schoten <- c(6, 18, 4, 3, 2, 2)
```

Wir überschreiben den ursprünglichen Datensatz "data" durch den erweiterten Datensatz

```
> data <- data
```

```
> data
```

	Pflanzen_Nr	fertile_Samen	abortierte_Samen	ausgezaehlte_Schoten
1	At1	377	10	6
2	At2	879	30	18
3	At3	216	41	4
4	At4	93	71	3
5	At5	98	22	2
6	At6	103	6	2

Aufgabe 1d:

Der Übersichtlichkeit halber sollen die Daten aufsteigend nach der Zahl der ausgezählten Schoten sortiert werden. Hierzu verwenden wir die Funktion order().

Die Funktion order(data$ausgezaehlte_Schoten) erstellt einen internen Vektor mit der Reihenfolge. Der wird dann als Index für die Sortierung der Reihen genutzt und auf den Datensatz "data" angewendet

```
> newdata2 <- data[order(data$ausgezaehlte_Schoten),]
```

```
> newdata2
```

	Pflanzen_Nr	fertile_Samen	abortierte_Samen	ausgezaehlte_Schoten
5	At5	98	22	2
6	At6	103	6	2
4	At4	93	71	3
3	At3	216	41	4
1	At1	377	10	6
2	At2	879	30	18

Aufgabe 2:

Für weitere Übungen zum Indizieren von Dataframes öffnen wir in R den Beispeildatensatz "iris" als Dataframe d.

Der Datensatz kann einfach durch Eingabe von "iris" abgerufen werden

Wir legen den Datensatz als dataframe "d" an

```
> d <- iris
```

```
> attach(iris)
```

Zur Sicherheit noch die Abfrage des Datentyps von "iris"

```
> class(iris)
```

```
[1] "data.frame"
```

Aufgabe 3:

Nun sehen wir mit head() die oberen Zeilen der Daten an. Danach möchten wir alle Zeilen Auslesen, die Daten für die Art "virginica" enthält.

Eine Möglichkeit zur Indizierung ist die Abfrage mit which(). which() generiert einen nummerischen Vektor, der alle Elemente enthält, die in dem entsprechenden logischen Vektor TRUE sind. Generieren Sie zunächst den nummerischen Vektor der Elemente. In einem zweiten Schritt erstellen Sie ein neues Dataframe "subset" aller Daten der Art "virginica".

Zunächst generieren wir den nummerischen Vektor der alle Elemente enthält

```
> which(d[,5] == "virginica")
 [1] 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
119
[20] 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137
138
[39] 139 140 141 142 143 144 145 146 147 148 149 150
```

Durch das Indizieren wird ein neues reduziertes Dataframe subset erstellt

```
> subset <-d[which(d[,5] == "virginica"), ]
> head(subset)
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
101          6.3         3.3          6.0         2.5 virginica
102          5.8         2.7          5.1         1.9 virginica
103          7.1         3.0          5.9         2.1 virginica
104          6.3         2.9          5.6         1.8 virginica
105          6.5         3.0          5.8         2.2 virginica
106          7.6         3.0          6.6         2.1 virginica
```

Aufgabe 4:

Erstellen Sie nun ein neues Dataframe "subset2" aus "subset", dass alle Daten der Art "virginica" enhält, für die die "Petal.Width" kleiner als 2 ist.

```
#  Die Indizierung kann ganz analog mit which() durchgeführt werden

> subset2 <- subset[which(subset[,4] < 2),]

> subset2
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
102	5.8	2.7	5.1	1.9	virginica
104	6.3	2.9	5.6	1.8	virginica
107	4.9	2.5	4.5	1.7	virginica
108	7.3	2.9	6.3	1.8	virginica
109	6.7	2.5	5.8	1.8	virginica
112	6.4	2.7	5.3	1.9	virginica
117	6.5	3.0	5.5	1.8	virginica
120	6.0	2.2	5.0	1.5	virginica
124	6.3	2.7	4.9	1.8	virginica
126	7.2	3.2	6.0	1.8	virginica
127	6.2	2.8	4.8	1.8	virginica
128	6.1	3.0	4.9	1.8	virginica
130	7.2	3.0	5.8	1.6	virginica
131	7.4	2.8	6.1	1.9	virginica
134	6.3	2.8	5.1	1.5	virginica
135	6.1	2.6	5.6	1.4	virginica
138	6.4	3.1	5.5	1.8	virginica
139	6.0	3.0	4.8	1.8	virginica
143	5.8	2.7	5.1	1.9	virginica
147	6.3	2.5	5.0	1.9	virginica
150	5.9	3.0	5.1	1.8	virginica

4. Übungen zu Schleifen

Aufgabe 1:

Nutzen Sie die ifelse() Funktion, um aus einem Vektor mit den ganzen Zahlen von 10 bis 20 "nicht richtig" auszugeben, wenn das Element ungleich 16 ist und "Hurra" wenn das Element 16 ist.

Hier ist die Lösung

```
> ifelse(x == 16, "Hurra", "nicht richtig")
 [1] "nicht richtig" "nicht richtig" "nicht richtig" "nicht richtig"
 [5] "nicht richtig" "nicht richtig" "Hurra"         "nicht richtig"
 [9] "nicht richtig" "nicht richtig" "nicht richtig"
```

Aufgabe 2:

Nutzen Sie repeat() und break für einen Countdown von 9 auf 0.

i wird zunächst als 1 definiert, dann jeweils um 1 reduziert bis der Wert 0 erreicht ist.

```
> i <- 10
> repeat{ i <- i-1
+ print(i)
+ if(i == 0) break}
[1] 9
[1] 8
[1] 7
[1] 6
[1] 5
[1] 4
[1] 3
[1] 2
[1] 1
[1] 0
```

Aufgabe 3:

Generieren Sie eine Schleife, um die Zahlen 1 bis 10 mit 3 zu multiplizieren und das Ergebnis auszugeben

Hier die Vorgehensweise und das Ergebnis

```
> x <- c(1:10)

> i <- 1

> for (i in x) print (i * 3)

[1] 3

[1] 6

[1] 9

[1] 12

[1] 15

[1] 18

[1] 21

[1] 24

[1] 27

[1] 30
```

5. Übungen zu besonderen Funktionen

Aufgabe 1:

Benutzen Sie die grep() Funktion, um aus folgendem Vektor alle Elemente auszulesen, in denen RKURS steht:

```
x <- c("KURSFREITAGHEUTE", "HEUTERKURSTOLL", "RKUSISTEIN",
"ALLERKURSESINDAUSGEBUCHT", "ALLESKURSIVODER")

# Nach Einlesen des Vektors können die entsprechenden Elemente
folgendermassen wieder ausgegeben werden.
> grep("RKURS", x, perl = TRUE, value = TRUE)
[1] "HEUTERKURSTOLL"          "ALLERKURSESINDAUSGEBUCHT"
```

Aufgabe 2:

Erstellen Sie folgende Matrix: x <- matrix(c(1:100), nrow = 10). Lesen Sie dann die kleinsten Zahlen in allen Spalten unter Verwendung von apply() aus.

```
# Zunächst erstellen wir die Matrix und sehen sie an
> x <- matrix(c(1:100), nrow = 10)
> x
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1   11   21   31   41   51   61   71   81    91
 [2,]    2   12   22   32   42   52   62   72   82    92
 [3,]    3   13   23   33   43   53   63   73   83    93
 [4,]    4   14   24   34   44   54   64   74   84    94
 [5,]    5   15   25   35   45   55   65   75   85    95
 [6,]    6   16   26   36   46   56   66   76   86    96
 [7,]    7   17   27   37   47   57   67   77   87    97
 [8,]    8   18   28   38   48   58   68   78   88    98
 [9,]    9   19   29   39   49   59   69   79   89    99
[10,]   10   20   30   40   50   60   70   80   90   100
```

Dann lesen wir die kleinsten Werte mit apply() aus

> apply(x, 2, min)

[1] 1 11 21 31 41 51 61 71 81 91

6. Übungen zu grafischen Darstellungen

Aufgabe 1:

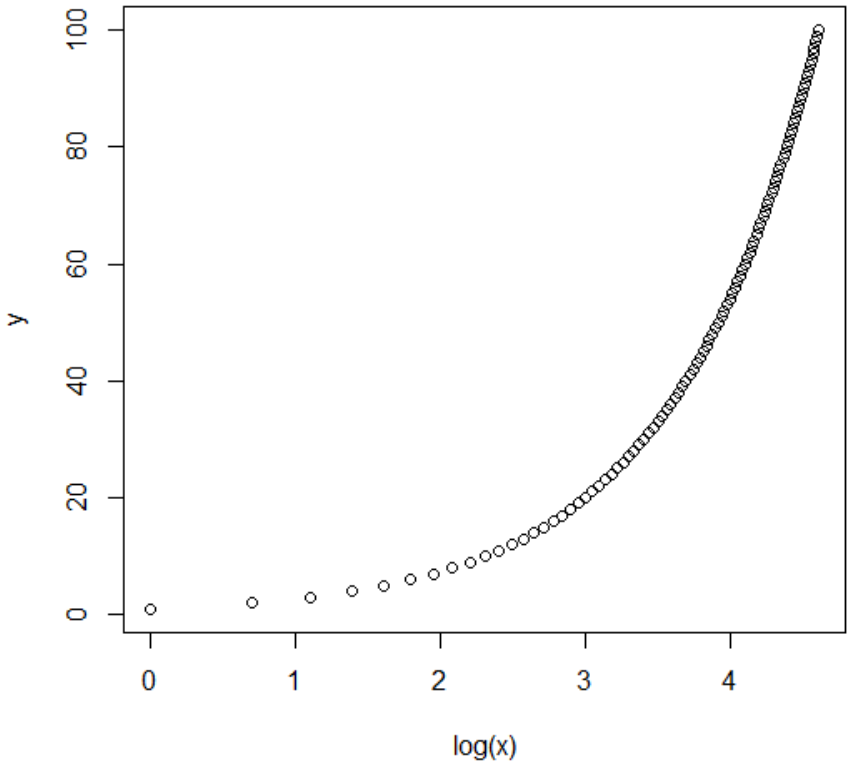
Erstellen Sie einen Scatterplot der Zahlen von log(1) bis log(100) auf der x-Achse gegen die Zahlen 1 bis 100 auf der y-Achse.

Zunächst erstellen wir die Vektoren x und y, dann wird der einfache Scatterplot mit dem Befehl plot() erstellt.

> x <- c(1:100)

> y <- c(1:100)

> plot(log(x), y)



Aufgabe 2:

Stellen Sie in einem Histogramm die Verteilung der Längen der Kronblätter der Art "setosa" aus dem Datensatz Iris dar.

Zunächst weisen wir den Iris Datensatz der Variablen x zu

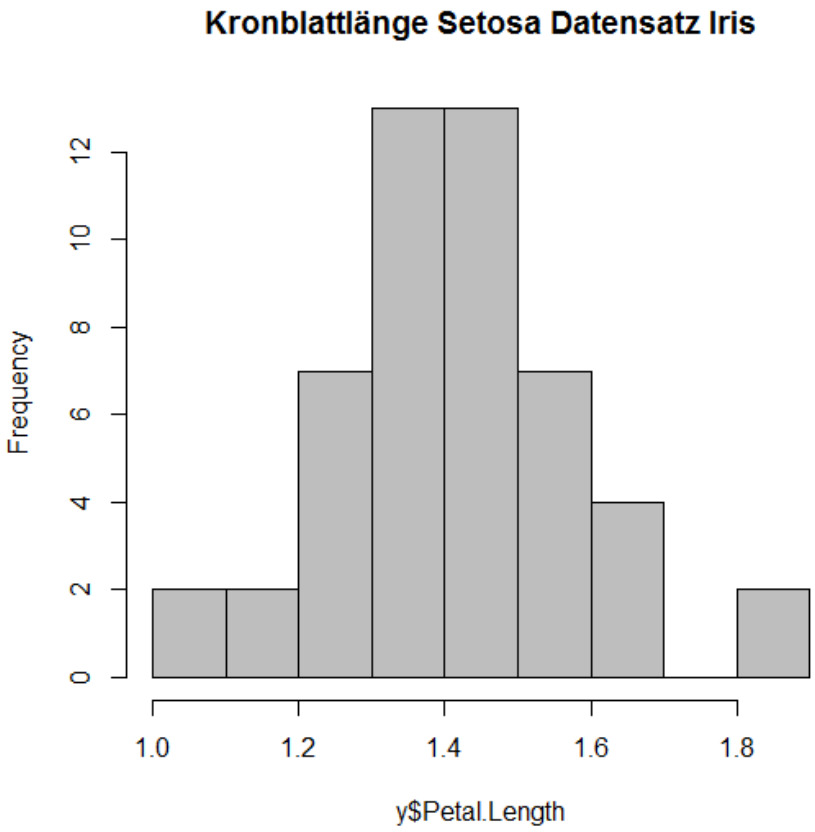
```
> x <- iris
```

Dann erstellen wir einen reduzierten Datensatz aller Daten von "setosa" unter der Variablen y.

```
> y <- x[which(x[,5] == "setosa"),]
```

Nun können wir mit hist() ein Histogram erstellen

```
> hist(y$Petal.Length, col = "gray", main = "Kronblattlänge Setosa Datensatz Iris")
```



Aufgabe 3:

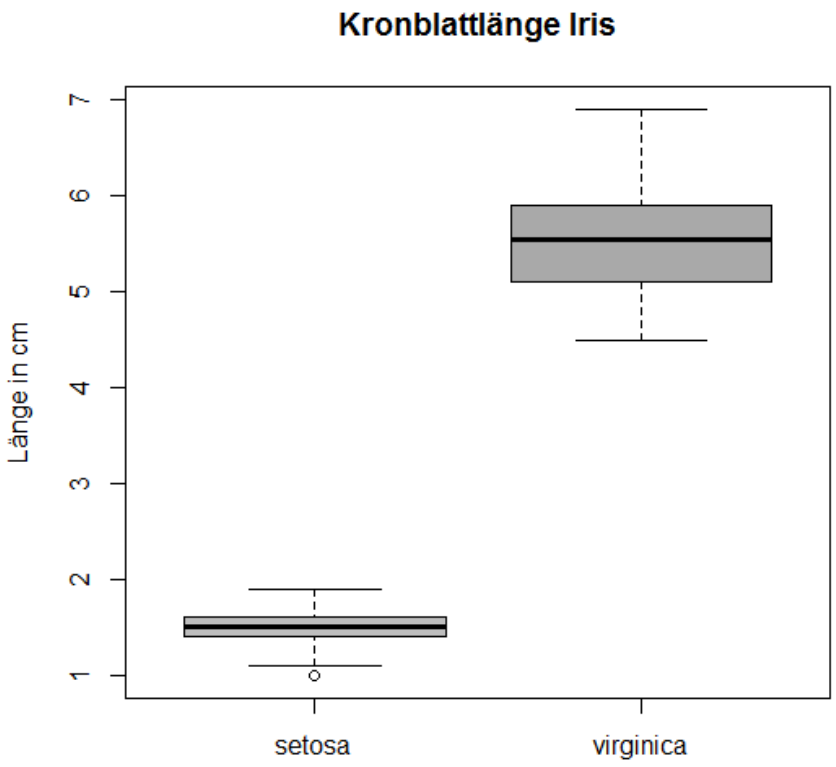
Erstellen Sie einen Box und Whiskers Plot von den Längen der Kronblätter von "setosa" und von "virginica".

Ein einfacher Weg ist es, auch hier einen reduzierten Datensatz für virginica zu bilden

> z <- x[which(x[,5] == "virginica"),]

Dann kann ein Boxplot z.B. folgendermassen erstellt werden

> boxplot(y$Petal.Length, z$Petal.Length, col = c("gray", "darkgray"), names = c("setosa", "virginica"), main = "Kronblattlänge Iris", ylab = "Länge in cm")



Aufgabe 4:

Plotten Sie die Mittelwerte der Längen der Kronblätter von "setosa" und "virginica" als Stabdiagramme (barplot()). Fügen Sie dem Balkendiagramm die Standardabweichungen für die Mittelwerte der Längen der Kronblätter von "setosa" und "virginica" als Fehlerbalken hinzu. Dazu benötigen Sie das Paket gplots und die Funktion barplot2(). Installieren und laden Sie zunächst gplots und sehen sich dann die Hilfe zu barplot2() an

Zunächst können wir einen Vektor "data" mit den Mittelwerten erstellen

> data <- c(mean(y$Petal.Length), mean(z$Petal.Length))

> barplot(data, col = c("orange", "darkgreen"), main = "Balkendiagramm", ylab = "in cm", names = c("setosa", "virginica"))

Laden des Paketes gplots

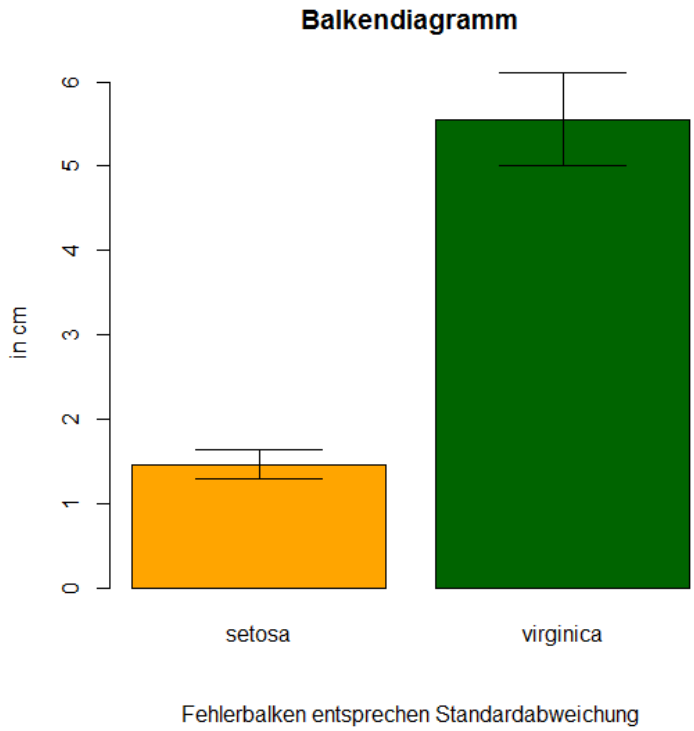
> library(gplots)

Die Standardabweichungen der Daten werden bestimmt

> fehler <- c(sd(y$Petal.Length), sd(z$Petal.Length))

Nun der entsprechende Barplot

> barplot2(data, col = c("orange", "darkgreen"), main = "Balkendiagramm", ylab = "in cm", names = c("setosa", "virginica"), sub = "Fehlerbalken entsprechen Standardabweichung", plot.ci = TRUE, ci.l = data-fehler, ci.u = data+fehler,)



7. Übungen zu Verteilungen und Hypothesentests

Aufgabe 1:

Prüfen Sie, ob die Mittelwerte der Längen der Kronblätter von "setosa" und "virginica" gleich oder signifikant unterschiedlich sind.

```
#  Hierzu wenden wir einen Studentischen t-test an
# Zunächst testen wir, ob die Varianz gleich ist

> var.test(y$Petal.Length, z$Petal.Length)

        F test to compare two variances

data:  y$Petal.Length and z$Petal.Length
F = 0.099016, num df = 49, denom df = 49, p-value = 1.875e-13
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.05618945 0.17448557
sample estimates:
ratio of variances
         0.0990164

> t.test(y$Petal.Length, z$Petal.Length)
```

```
# Wir testen mit einem Welch t-test
        Welch Two Sample t-test

data:  y$Petal.Length and z$Petal.Length
t = -49.986, df = 58.609, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -4.253749 -3.926251
sample estimates:
mean of x mean of y
    1.462     5.552
```

Aufgabe 2:

Ausgehend vom Iris Datensatz möchten wir nun wissen, ob die Kelchblattbreiten bei "virginica" signifikant höher sing als bei "setosa". Zur Sicherheit testen wir auch, ob die Kelchblattbreiten von "virginica" signifikant kleiner sind als von "setosa". Erstellen Sie dazu die Teststatistik. Veranschaulichen Sie die Unterschiede grafisch.

```
# Zunächst wieder ein Test auf gleiche Varianzen
> var.test(y$Sepal.Width, z$Sepal.Width)

        F test to compare two variances

data:  y$Sepal.Width and z$Sepal.Width
F = 1.3816, num df = 49, denom df = 49, p-value = 0.2614
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.7840128 2.4346017
sample estimates:
ratio of variances
          1.381578
```

Hier haben wir es mit gleichen Varianzen zu tun und wir können im folgenden t-Test var.equal = TRUE definieren

Nun lassen wir einen einseitigen t-Test folgen

```
> t.test(y$Sepal.Width, z$Sepal.Width, alternative = c("greater"), var.equal = TRUE)

        Two Sample t-test

data:  y$Sepal.Width and z$Sepal.Width
t = 6.4503, df = 98, p-value = 2.123e-09
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 0.3371241       Inf
sample estimates:
mean of x mean of y
    3.428     2.974
```

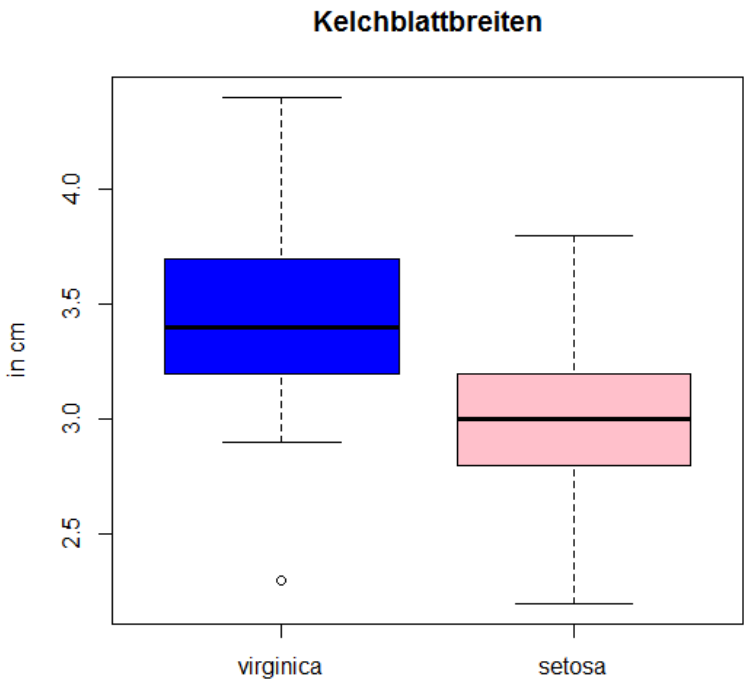
Das Ergebnis besagt, dass die Kelchblattbreite von "virginica" signifikant größer ist als von "setosa".

Nun der Gegentest, ob die Kelchblattbreite von "virginica" signifikant kleiner ist als von "setosa"

```
> t.test(y$Sepal.Width, z$Sepal.Width, alternative = c("less"), var.equal = TRUE)

        Two Sample t-test

data:  y$Sepal.Width and z$Sepal.Width
t = 6.4503, df = 98, p-value = 1
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf 0.5708759
sample estimates:
mean of x mean of y
    3.428     2.974
```

Grafische Veranschaulichung durch einen Boxplot

```
> boxplot(y$Sepal.Width, z$Sepal.Width, col = c("blue", "pink"), main =
"Kelchblattbreiten", ylab = "in cm", names = c("virginica", "setosa"))
```



Aufgabe 3:

Im Labor untersuchen Sie mutante Arabidopsis Linien mit einem Defekt während der Samenentwicklung. Durch Öffnen der Schoten konnten Sie alle normal entwickelten Samen auszählen (norm), sowie abortierte Samen (abort). Von der Linie haben Sie jeweils 4 Schoten von 3 Pflanzen ausgezählt und die Daten in der Tabelle seedabort.txt hinterlegt.

Lesen Sie zunächst die Daten in R ein und sehen Sie an. Sie möchten herausfinden, ob es sich um einen gametophytischen oder einen zygotischen Defekt handelt. Ihre Erwartungen sind, dass bei einem gametophytischen Defekt das Verhältnis von norm : abort gleich 1 : 1 ist. Bei einem zygotischen Defekt erwarten Sie das Verhältnis norm : abort von 3 : 1. Testen Sie beide Szenarien mit einer geeigneten Teststatistik.

Einlesen und Ansehen der Daten

```
> data = read.table(file = "seedabort.txt", header = TRUE)
> data
                  norm abort
Pflanze1_Schote1   30    23
Pflanze1_Schote2   24    29
Pflanze1_Schote3   18    34
Pflanze1_Schote4   25    26
Pflanze2_Schote1   31    27
Pflanze2_Schote2   19    25
Pflanze2_Schote3   27    28
Pflanze2_Schote4   24    22
Pflanze3_Schote1   26    23
Pflanze3_Schote2   30    19
Pflanze3_Schote3   26    24
Pflanze3_Schote4   23    30
```

```
# Dann benötigen wir die Summe der Auszählungen
> sum(data$norm)
[1] 303
> sum(data$abort)
[1] 310

# Nun die entsprechende Matrix
> count <- matrix(c(303, 310), nrow = 2)

data:  count
X-squared = 0.2397, df = 1, p-value = 0.6244

> count <- matrix(c(303, 310), nrow = 2)

# Test auf einen gametophytischen Defekt
> chisq.test(count, p = c(1/2, 1/2))

        Chi-squared test for given probabilities

data:  count
X-squared = 0.079935, df = 1, p-value = 0.7774

# Test auf einen zygotischen Defekt
> chisq.test(count, p = c(3/4, 1/4))

        Chi-squared test for given probabilities

data:  count
X-squared = 213.77, df = 1, p-value < 2.2e-16
```

8. Übungen zur linearen Regression

Aufgabe 1:

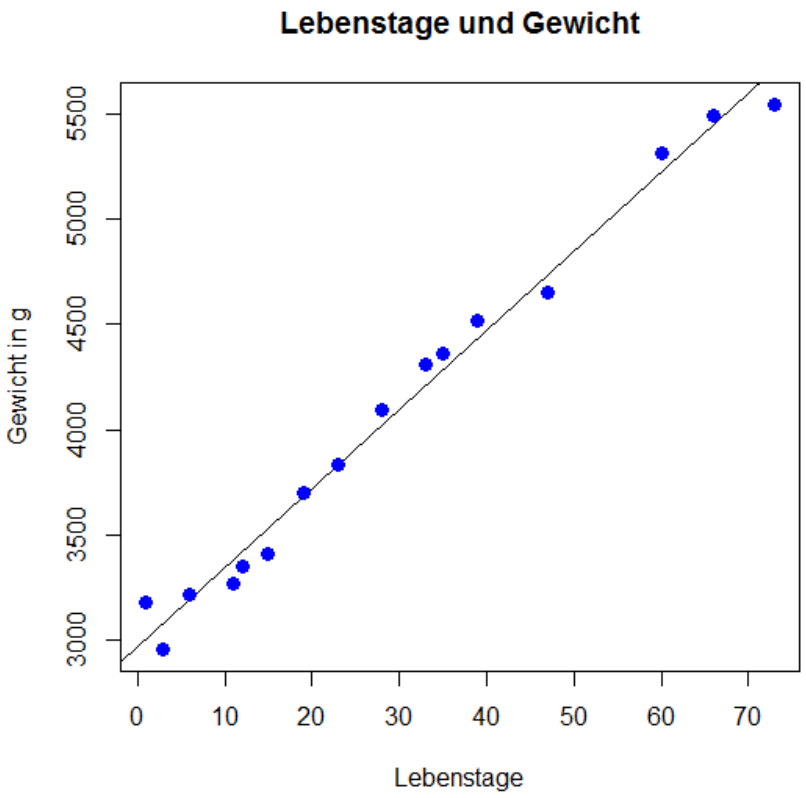
Hier bedienen wir uns eines Beispieldatensatzes von (https://de.wikibooks.org/wiki/GNU_R:_Anwendungsbeispiele).

Der Datensatz wird folgendermaßen erzeugt:

```
> x <- c(1, 3, 6, 11, 12, 15, 19, 23, 28, 33, 35, 39, 47, 60, 66, 73)

> y <- c(3180, 2960, 3220, 3270, 3350, 3410, 3700, 3830, 4090, 4310, 4360, 4520,
4650, 5310, 5490, 5540)

> bsp5 <- data.frame(x,y)

> colnames(bsp5) <- c("Lebenstag", "Gewicht")

> bsp5

   Lebenstag Gewicht
1          1    3180
2          3    2960
3          6    3220
4         11    3270
5         12    3350
6         15    3410
7         19    3700
8         23    3830
9         28    4090
10        33    4310
11        35    4360
12        39    4520
13        47    4650
14        60    5310
15        66    5490
16        73    5540
```

Erstellen Sie einen Scatterplot inklusive Regressionslinie.

> plot(x, y, col = "blue", main = "Lebenstage und Gewicht", abline(lm(y ~ x)),cex = 1.3,pch = 16,xlab = "Lebenstage",ylab = "Gewicht in g")



9. Übungen zu Verteilungen

Aufgabe 1a:

Ziehen Sie 1000x zufällig fünf Zahlen zwischen 1 und 6 und berechnen Sie die Mittelwerte. Plotten Sie ein Histogramm der Mittelwerte.

```
# Ziehen von 5 Zahlen zwischen 1 und 6 und generieren der Mittelwerte, 1000 Wiederholungen
> means <- numeric(1000)
> for (i in 1:1000){
+   means[i]<-mean(runif(5)*6)
+   }

# Plotten des Histogramms der Mittelwerte
> hist(means, ylim = c(0, 280))
```

Aufgabe 1b:

Ermitteln Sie den Mittelwert der Mittelwerte und die Standardabweichung.

```
> mean(means)
[1] 2.988968
> sd(means)
[1] 0.779267
```

Aufgabe 1c:

Plotten Sie die Kurve der Normalverteilung über in das Histogramm

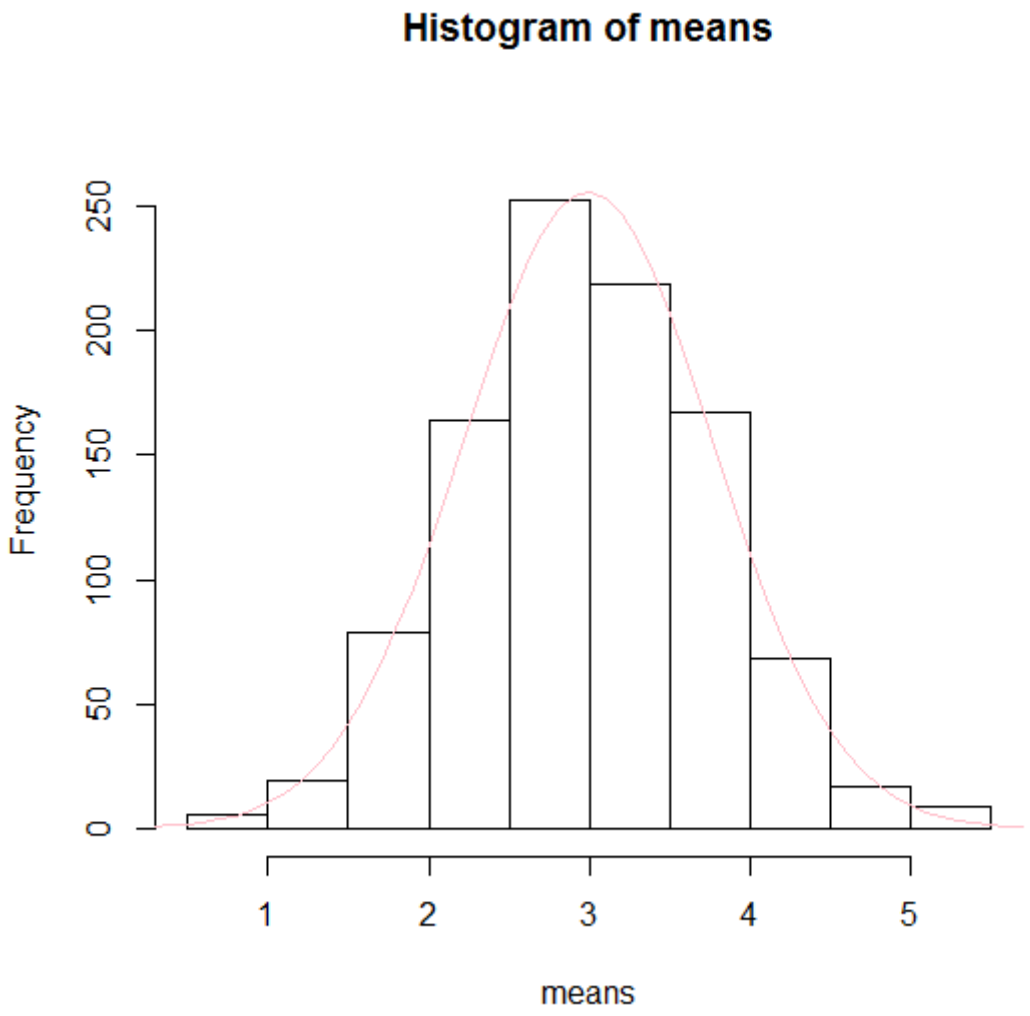
Wir generieren eine Sequenz von Werten zwischen 0 und 6 die sich jeweils um 0.1 erhöhen für die x-Achse

> xwerte <- seq(0, 6, 0.1)

Wir geben den Mittelwert und die Standardabweichung für Dichteverteilung auf der y-Achse vor

> ywerte <- dnorm(xwerte, mean = 2.988968, sd = 0.779267) * 500

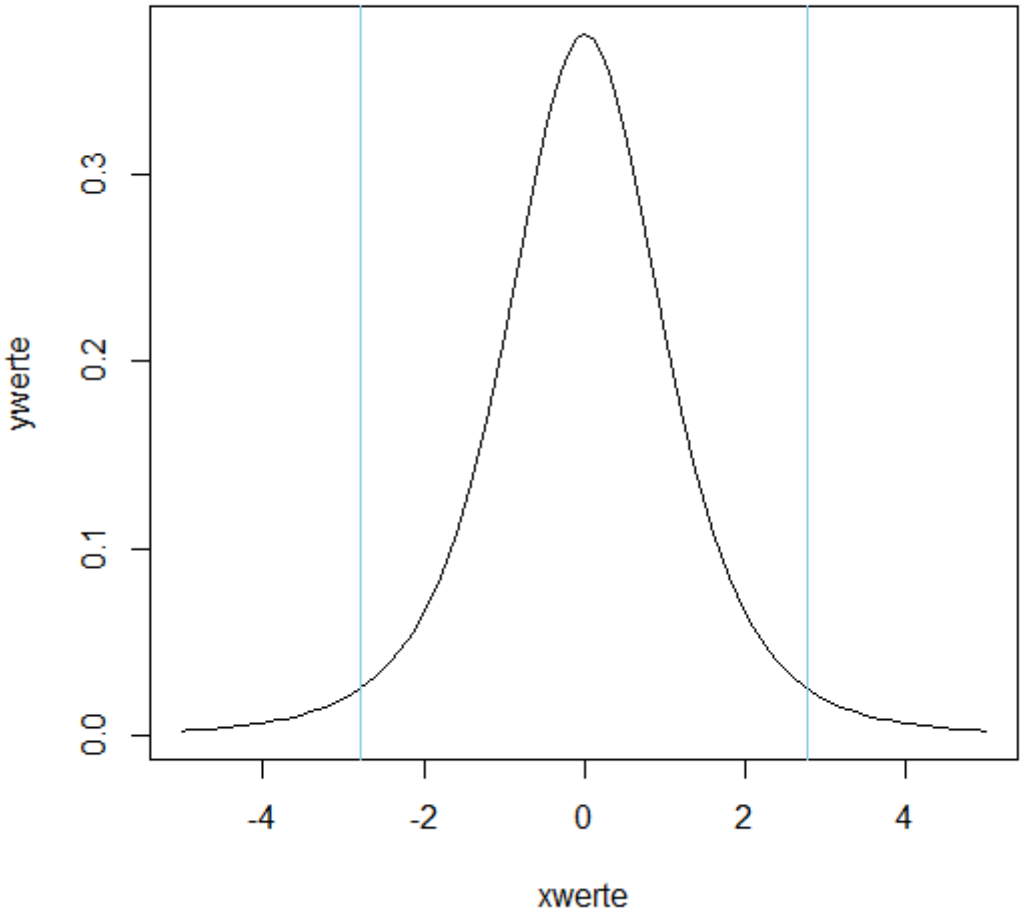
> lines(xwerte, ywerte, col = "pink")



Aufgabe 2:

Plotten Sie eine Kurve der t-Verteilung für 4 Freiheitsgrade in den Grenzen zwischen ± 5 Freiheitsgraden. Zeichnen Sie im Anschluss das Intervall ein, in das 95% aller Werte fallen.

```
# Generieren einer Sequenz von Werten
> xwerte <- seq(-5, 5, 0.1)

# Aufrufen der t-Verteilung für 4 Freiheitsgrade
> ywerte <- dt(xwerte, df = 4)

# Der Plot kann noch angepasse werden bezüglich Beschriftungen usw
> plot(xwerte, ywerte, type = "l", lty = 1)

# Bestimmen des Konfidenzintervalls
> qt(c(0.025, 0.975), df = 4)
[1] -2.776445  2.776445

# Eintragen der Linien in den Plot
> abline(col = "skyblue", v = -2.776445)
> abline(col = "skyblue", v = 2.776445)
```


ywerte
0.0
0.1
0.2
0.3
-4
-2
0
2
4
xwerte