Dr. Katharina Best
Sommersemester 2011

Kurze Einführung in R

WiMa-Praktikum

Erste Schritte

- R wird durch Eintippen von 'R' in der Konsole gestartet.
- Beendet wird es durch q() oder quit().
- Es existieren auch integrierte Lösungen mit Editoren.
- Die interessanteste Variante ist ESS – Emacs Speaks Statistics
- Hilfe findet sich unter

```
    help.start();
    help(sum);
    ?sum;
    example(mean);
```

- Zuweisung von Zahlen

```
> x <- 3
> y <- exp(x) - 2 * 3 + 4/5
```

- Ergebnis ansehen

```
> y

[1] 14.88554
```

- Vektoren bilden

```
> x <- c(1, 7, 4, 2, 5)
> y <- exp(x) - 2
> y

[1]    0.7182818 1094.6331584   52.5981500    5.3890561  146.4131591
```

```
> y[2] <- 1
> y

[1]   0.7182818   1.0000000  52.5981500   5.3890561 146.4131591
```

- Folgen bilden

```
> rep(1, 5)

[1] 1 1 1 1 1

> rep(1:5, times = 2)

 [1] 1 2 3 4 5 1 2 3 4 5

> rep(1:5, each = 2)

 [1] 1 1 2 2 3 3 4 4 5 5

> rep(1:5, length = 2)

[1] 1 2

> 1:5

[1] 1 2 3 4 5

> seq(1, 5)

[1] 1 2 3 4 5

> seq(1, 17, by = 4)

[1]  1  5  9 13 17

> seq(1, 5, by = 0.5)

[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

> 2:10/2

[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

> seq(1, 20, length = 5)

[1]  1.00  5.75 10.50 15.25 20.00
```

- Auf Elemente von Vektoren zugreifen

```
> x <- 1:100/10
> x[1:10]

 [1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

> x[c(1, 10, 15)]

[1] 0.1 1.0 1.5

> x[c(10, 1, 15)]

[1] 1.0 0.1 1.5

> x[x > 4.5]

 [1]  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.9  6.0
[16]  6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9  7.0  7.1  7.2  7.3  7.4  7.5
[31]  7.6  7.7  7.8  7.9  8.0  8.1  8.2  8.3  8.4  8.5  8.6  8.7  8.8  8.9  9.0
[46]  9.1  9.2  9.3  9.4  9.5  9.6  9.7  9.8  9.9 10.0

> x > 4.5

  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
 [49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [73]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [85]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [97]  TRUE  TRUE  TRUE  TRUE

> x[-(1:10)]

 [1]  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5
[16]  2.6  2.7  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.0
[31]  4.1  4.2  4.3  4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4  5.5
[46]  5.6  5.7  5.8  5.9  6.0  6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9  7.0
[61]  7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9  8.0  8.1  8.2  8.3  8.4  8.5
[76]  8.6  8.7  8.8  8.9  9.0  9.1  9.2  9.3  9.4  9.5  9.6  9.7  9.8  9.9 10.0
```

- Vektoren sortieren

```
> x <- c(3, 2, 5, 1, 7)
> y <- c(1, 1, 4, 6, 2)
> sort(x)

[1] 1 2 3 5 7

> order(x)
```

```
[1] 4 2 1 3 5
```

```
> x[order(x)]
```

```
[1] 1 2 3 5 7
```

```
> o <- order(x)
> x[o]
```

```
[1] 1 2 3 5 7
```

```
> y[o]
```

```
[1] 6 1 1 4 2
```

- Vektoren mit Zeichenfolgen

```
> c(rep("ill", 10), rep("healthy", 10))
```

```
 [1] "ill"     "ill"     "ill"     "ill"     "ill"     "ill"     "ill"
 [8] "ill"     "ill"     "ill"     "healthy" "healthy" "healthy" "healthy"
[15] "healthy" "healthy" "healthy" "healthy" "healthy" "healthy"
```

```
> x <- 10
> name <- "The object x"
> cat(name, "equals", x, "\n")
```

```
The object x equals 10
```

- Informationen über Vektoren

```
> length(rnorm(10))
```

```
[1] 10
```

```
> unique(c(rep("ill", 10), rep("healthy", 10)))
```

```
[1] "ill"     "healthy"
```

- Matrizen

```
> x <- c(1, 2, 3, 4)
> matrix(x, nrow = 2, ncol = 2)
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```
> matrix(x, 2, 2, byrow = TRUE)
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4

> diag(3)

     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1

> diag(x = 10, nrow = 3, ncol = 4)

     [,1] [,2] [,3] [,4]
[1,]   10    0    0    0
[2,]    0   10    0    0
[3,]    0    0   10    0

> A <- matrix(1:16, 4)
> B <- matrix(1:2, 4, 4)
```

- Matrizenoperationen

```
> A + B

     [,1] [,2] [,3] [,4]
[1,]    2    6   10   14
[2,]    4    8   12   16
[3,]    4    8   12   16
[4,]    6   10   14   18

> A * B

     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    4   12   20   28
[3,]    3    7   11   15
[4,]    8   16   24   32

> A %*% B

     [,1] [,2] [,3] [,4]
[1,]   46   46   46   46
[2,]   52   52   52   52
[3,]   58   58   58   58
[4,]   64   64   64   64

> A %x% diag(2)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    0    5    0    9    0   13    0
[2,]    0    1    0    5    0    9    0   13
[3,]    2    0    6    0   10    0   14    0
[4,]    0    2    0    6    0   10    0   14
[5,]    3    0    7    0   11    0   15    0
[6,]    0    3    0    7    0   11    0   15
[7,]    4    0    8    0   12    0   16    0
[8,]    0    4    0    8    0   12    0   16
```

```
> diag(2) %x% A
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    5    9   13    0    0    0    0
[2,]    2    6   10   14    0    0    0    0
[3,]    3    7   11   15    0    0    0    0
[4,]    4    8   12   16    0    0    0    0
[5,]    0    0    0    0    1    5    9   13
[6,]    0    0    0    0    2    6   10   14
[7,]    0    0    0    0    3    7   11   15
[8,]    0    0    0    0    4    8   12   16
```

- Auf Werte in Matrizen und Informationen zugreifen

```
> dim(B)
```

```
[1] 4 4
```

```
> A[, 1]
```

```
[1] 1 2 3 4
```

```
> A[1, ]
```

```
[1]  1  5  9 13
```

```
> A[3, 1]
```

```
[1] 3
```

- Höher dimensionale Arrays

```
> x <- array(1:27, dim = c(3, 3, 3))
> x[1, , ]
```

```
     [,1] [,2] [,3]
[1,]    1   10   19
[2,]    4   13   22
[3,]    7   16   25
```

```
> x[1, 2, ]

[1]  4 13 22
```

- Lists

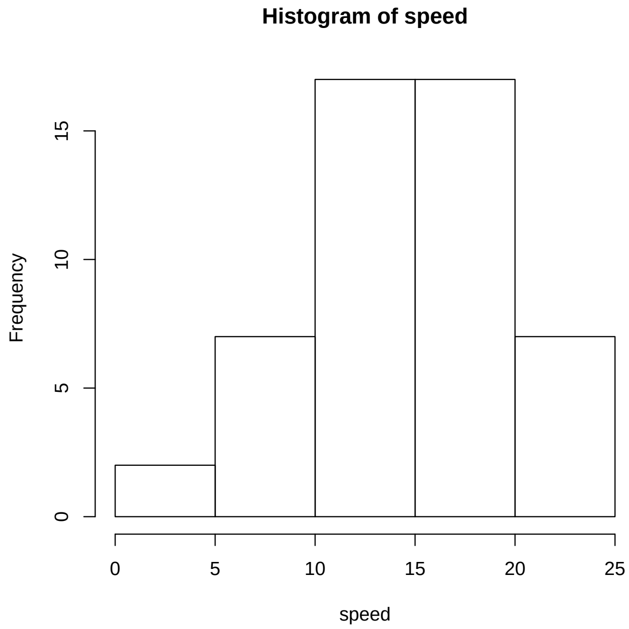
```
> course.info <- list(students = c("Sandra", "Karl", "Thomas",
+     "Nadine"), nr.of.exercises = 100, rooms = c("Multimedia Room",
+     "Seminar Room"))
> course.info$rooms

[1] "Multimedia Room" "Seminar Room"

> course.info$students[2]

[1] "Karl"
```

- Zufallsvariablen und Verteilungen

```
> rnorm(10)

 [1] -1.05283202  1.03829966 -1.19893243 -0.85100357  0.28439932 -1.13938751
 [7] -0.31076647 -0.52582979  0.22304642 -0.07904784

> rnorm(10, mean = 5, sd = 2)

 [1] 2.875791 4.708124 6.113218 4.724995 5.543958 6.838029 4.609193 6.011188
 [9] 6.762374 4.654707

> qnorm(0.975)

[1] 1.959964

> pnorm(0)

[1] 0.5

> dnorm(0)

[1] 0.3989423
```

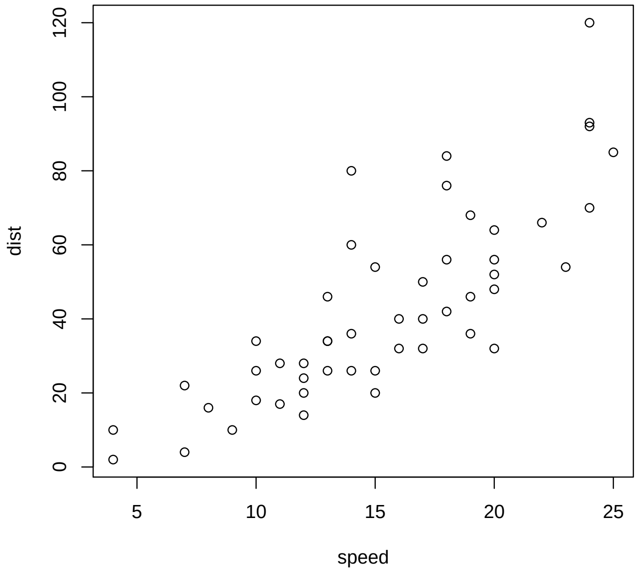
- Vorhandene Datensätze nutzen

```
> data(cars)
> attach(cars)
```

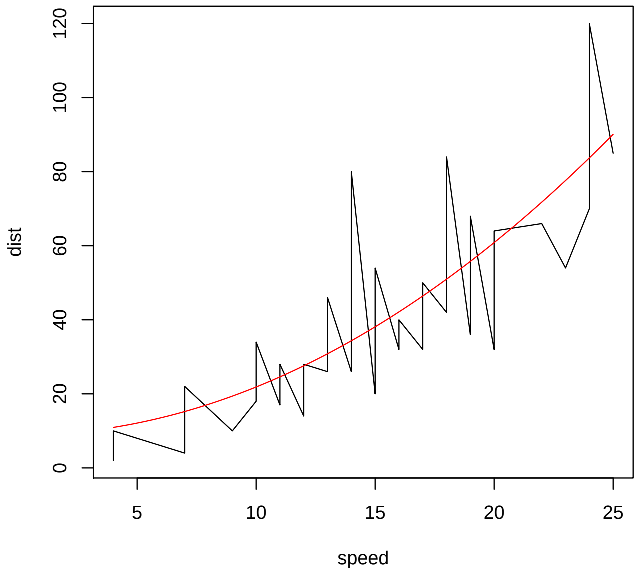


- Graphik

```
> plot(speed, dist)
```

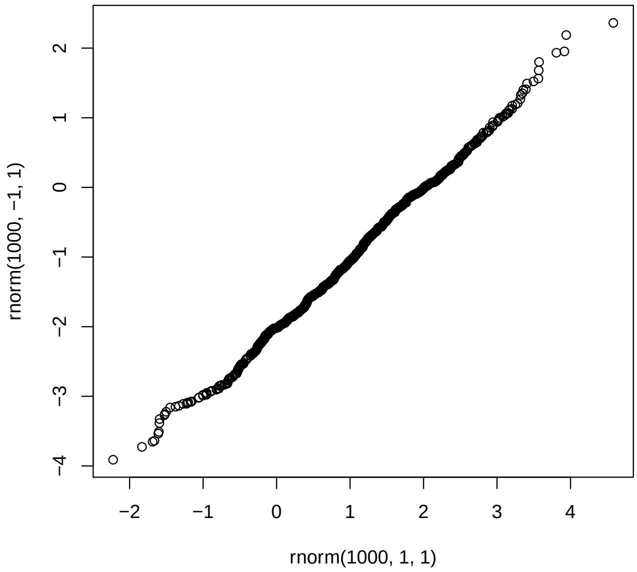


```
> plot(speed, dist, type = "l")
> abline(a = mean(dist), b = sd(dist))
> curve(8.86 + 0.13 * x^2, add = TRUE, col = "red")
```

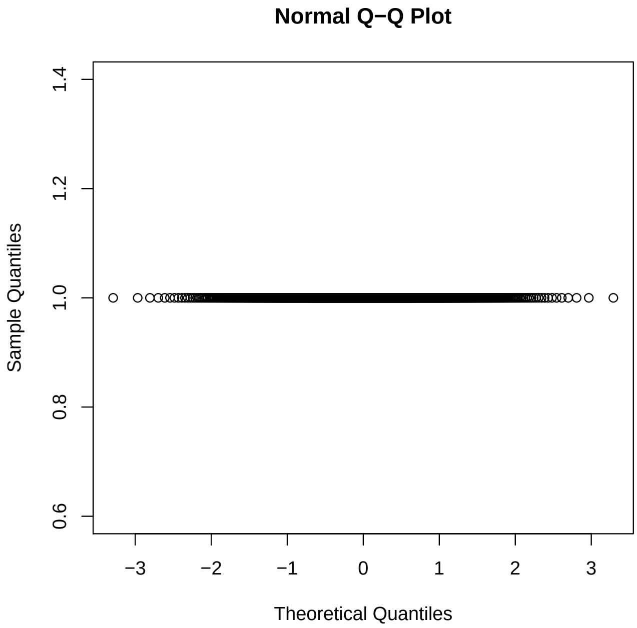


- Probability Plots

```
> qqplot(rnorm(1000, 1, 1), rnorm(1000, -1, 1))
```



```
> qqnorm(rnorm(1000, 1, 0))
```



- Eigene Funktionen definieren

```
> sum.of.squares <- function(x) {
+     sum(x^2)
+ }
> sum.of.squares(c(1, 3, 5, 7))

[1] 84
```

- Schleifen

```
> for (i in 1:10) {
+     print(i^2)
+ }

[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
[1] 49
[1] 64
[1] 81
[1] 100

> for (name in c("I", "want", "to", "go", "home!")) {
+     cat(name, "\n")
+ }
```

```
I
want
to
go
home!

> while (rnorm(1) < 1) {
+     print("hallo")
+ }

[1] "hallo"
[1] "hallo"
[1] "hallo"
```

Zu beachten ist, dass `R` Schleifen sehr langsam abarbeitet (wie `Matlab`). Deswegen sollte man bei wiederkehrenden Berechnungen diese nach Möglichkeit direkt auf Matrizen und Arrays anwenden. Dazu bietet `R` die Funktionen `apply()`, `mapply()`, `sapply()` und `tapply()`.

- Bedingungen

  ```
  > if (rnorm(1) > 0) {
  +     print("rnorm generated a positive value.")
  + } else {
  +     print("rnorm generated a negative value.")
  + }

  [1] "rnorm generated a positive value."
  ```

- Wo bin ich und was habe ich gemacht?

  Bisher benutzte Befehle können mit `history(max.show = 10)` eingesehen werden.

  ```
  > savehistory(file = "first.session.Rhistory")
  > getwd()

  [1] "~/katharina/Dokumente/WiMaPraktikum"
  ```

- Pakete verwenden Auflistung aller installierten Pakete mit `library()`.

  ```
  > library(Kendall)
  > Kendall(rnorm(20), rnorm(20))

  tau = -0.116, 2-sided pvalue =0.49566
  ```

  Neue Pakete können von `CRAN` heruntergeladen werden.

- Externe Dateien nutzen

  ```
  > my.file <- file("test.r", "w")
  > cat("test <- function(x) {print(x)}", "\n", file = my.file)
  ```

```
> close(my.file)
> source("test.r")
> test

function (x)
{
    print(x)
}
```

Und vor allem: Viel Spaß!