

Einführung ins Statistikprogramm



A. Mändle

Institut für Statistik, Universität Bremen

WS-2019/20



Andreas Mändle

Raum: Linzerstr. 4, Raum 41070

Tel: (218) 63782

Mail: maendle@uni-bremen.de

Der Kursaufbau und die Unterlagen orientieren sich am Folienskript von Dr. Farhad Arzideh, der den Kurs bis WS 2017/18 durchgeführt hat.

Inhalte I

Organisatorisches


0.1 Termine, Übungen	5
0.2 Bezug und Installation	14
0.3 Lehrmaterial und Literatur	15
0.4 Lernziel	17
0.5 Ausblick: Beispiele aus der Vorlesung	20

1. Einstieg und Grundlegendes zu 42

1.1 Hintergrund	43
1.2 Grundlegende Bedienung	56
1.3 Dokumentation und Hilfesystem	75
1.4 Mathematische Operatoren	80
1.5 Logische Operatoren	81
1.6 Vektoren	84
1.7 Funktionen	85

2. Datentypen und Datenimport 92

Inhalte II

2.1	Datentypen	93
2.2	Datenstrukturen	94
2.3	Kontrollstrukturen	148
2.4	Eingabe/Einlesen und Ausgabe von Daten	182
3.	Deskriptive Statistik mit <i>R</i>	193
4.	Verteilungen & Zufallszahlen in 	349
5.	Schliessende Statistik: Testen und Schätzen	441
5.1	Binomialtest	446
5.2	t-Test	502
5.3	Multiples Testen	549
5.4	F-Test	572
5.5	Anpassungstests	582
5.6	Testen auf Unabhängigkeit	603
5.7	Varianzanalyse	618
5.8	Varianzanalyse und Regressionsanalyse	647
5.9	Regressionsanalyse	648

Termine, Übungen

Raum	Zeit
Linzerstr. 4, Raum 040010	montags 13:00-16:00 Uhr

- Übungen: Präsenz- und Hausaufgaben (PA, HA)
- Abgabe der HA: wann?
- Bearbeitung der HA möglichst zu zweit
- Zum Bestehen des Kurses:
 - 1 regelmäßige und aktive Teilnahme am Kurs
 - 2 PA bearbeiten
 - 3 75% der HA sinnvoll bearbeiten
- Benötigt jemand eine Note?
(dann evtl. Zusatzleistung)
- Bringen Sie Ihren Laptop mit

Termine, Übungen

Raum	Zeit
Linzerstr. 4, Raum 040010	montags 13:00-16:00 Uhr

- Übungen: Präsenz- und Hausaufgaben (PA, HA)
- Abgabe der HA: wann?
- Bearbeitung der HA möglichst zu zweit
- Zum Bestehen des Kurses:
 - 1 regelmäßige und aktive Teilnahme am Kurs
 - 2 PA bearbeiten
 - 3 75% der HA sinnvoll bearbeiten
- Benötigt jemand eine Note?
(dann evtl. Zusatzleistung)
- Bringen Sie Ihren Laptop mit

Termine, Übungen

Raum	Zeit
Linzerstr. 4, Raum 040010	montags 13:00-16:00 Uhr

- Übungen: Präsenz- und Hausaufgaben (PA, HA)
- Abgabe der HA: wann?
- Bearbeitung der HA möglichst zu zweit
- Zum Bestehen des Kurses:
 - 1 regelmäßige und aktive Teilnahme am Kurs
 - 2 PA bearbeiten
 - 3 75% der HA sinnvoll bearbeiten
- Benötigt jemand eine Note?
(dann evtl. Zusatzleistung)
- Bringen Sie Ihren Laptop mit

Termine, Übungen

Raum	Zeit
Linzerstr. 4, Raum 040010	montags 13:00-16:00 Uhr

- Übungen: Präsenz- und Hausaufgaben (PA, HA)
- Abgabe der HA: wann?
- Bearbeitung der HA möglichst zu zweit
- Zum Bestehen des Kurses:
 - 1 regelmäßige und aktive Teilnahme am Kurs
 - 2 PA bearbeiten
 - 3 75% der HA sinnvoll bearbeiten
- Benötigt jemand eine Note?
(dann evtl. Zusatzleistung)
- Bringen Sie Ihren Laptop mit

Termine, Übungen

Raum	Zeit
Linzerstr. 4, Raum 040010	montags 13:00-16:00 Uhr

- Übungen: Präsenz- und Hausaufgaben (PA, HA)
- Abgabe der HA: wann?
- Bearbeitung der HA möglichst zu zweit
- Zum Bestehen des Kurses:
 - 1** regelmäßige und aktive Teilnahme am Kurs
 - 2 PA bearbeiten
 - 3 75% der HA sinnvoll bearbeiten
- Benötigt jemand eine Note?
(dann evtl. Zusatzleistung)
- Bringen Sie Ihren Laptop mit

Termine, Übungen

Raum	Zeit
Linzerstr. 4, Raum 040010	montags 13:00-16:00 Uhr

- Übungen: Präsenz- und Hausaufgaben (PA, HA)
- Abgabe der HA: wann?
- Bearbeitung der HA möglichst zu zweit
- Zum Bestehen des Kurses:
 - 1** regelmäßige und aktive Teilnahme am Kurs
 - 2** PA bearbeiten
 - 3** 75% der HA sinnvoll bearbeiten
- Benötigt jemand eine Note?
(dann evtl. Zusatzleistung)
- Bringen Sie Ihren Laptop mit

Termine, Übungen

Raum	Zeit
Linzerstr. 4, Raum 040010	montags 13:00-16:00 Uhr

- Übungen: Präsenz- und Hausaufgaben (PA, HA)
- Abgabe der HA: wann?
- Bearbeitung der HA möglichst zu zweit
- Zum Bestehen des Kurses:
 - 1** regelmäßige und aktive Teilnahme am Kurs
 - 2** PA bearbeiten
 - 3** 75% der HA sinnvoll bearbeiten
- Benötigt jemand eine Note?
(dann evtl. Zusatzleistung)
- Bringen Sie Ihren Laptop mit

Termine, Übungen

Raum	Zeit
Linzerstr. 4, Raum 040010	montags 13:00-16:00 Uhr

- Übungen: Präsenz- und Hausaufgaben (PA, HA)
- Abgabe der HA: wann?
- Bearbeitung der HA möglichst zu zweit
- Zum Bestehen des Kurses:
 - 1 regelmäßige und aktive Teilnahme am Kurs
 - 2 PA bearbeiten
 - 3 75% der HA sinnvoll bearbeiten
- Benötigt jemand eine Note?
(dann evtl. Zusatzleistung)
- Bringen Sie Ihren Laptop mit


Termine, Übungen

Raum	Zeit
Linzerstr. 4, Raum 040010	montags 13:00-16:00 Uhr

- Übungen: Präsenz- und Hausaufgaben (PA, HA)
- Abgabe der HA: wann?
- Bearbeitung der HA möglichst zu zweit
- Zum Bestehen des Kurses:
 - 1 regelmäßige und aktive Teilnahme am Kurs
 - 2 PA bearbeiten
 - 3 75% der HA sinnvoll bearbeiten
- Benötigt jemand eine Note?
(dann evtl. Zusatzleistung)
- Bringen Sie Ihren Laptop mit

Bezug und Installation

Original-Download von **R** unter <https://cran.r-project.org>

- Mac und Windows-Benutzer downloaden die Binaries/Installer.
- Hilfe zum Kompilieren von  (v.a. für Linux-Nutzer) unter <https://cran.r-project.org/doc/manuals/R-admin.html>

mögliche Alternative: z.B. *R Open* (Microsoft) unter <https://mran.microsoft.com/open>

Dringende Empfehlung: Zusätzlich zu R bzw. R Open die Entwicklungsumgebung **R Studio** installieren. Download für Windows, Mac und Linux unter <https://www.rstudio.com/products/rstudio/download/#download>

Optional: Für Nutzer der Editoren Emacs und WinEdt gibt es entsprechende Erweiterungen (<https://ess.r-project.org> ESS bzw. <https://cran.r-project.org/web/packages/RWinEdt/vignettes/RWinEdt.pdf> R-WinEdt). Windows-Nutzer können auch <https://sourceforge.net/p/tinn-r/wiki/Home/> Tinn-R ausprobieren.

Lehrmaterial und Literatur I

Folien, Aufgaben, Daten und Musterlösungen in:
StudIP: Einführung in die statistische Software R/Dateien/



M. Kohl,
Einführung in die statistische Datenanalyse mit R, 2015,



T. Rahlf,
Data Visualisation with R, 2017,



S. Stowell,
Using R for Statistics, 2014,



U. Ligges,
Programmieren mit R, 2008,
<http://link.springer.com/book/10.1007%2F978-3-540-79998-6>



M.J. Crawley,
Statistik mit R, 2012



L. Sachs, J. Hedderich,
Angewandte Statistik, Methodensammlung mit R 2006



D. Chen, K. Peace,
Clinical Trial Data Analysis Using R, 2011



D. Dubravko,
Statistik mit R, 2004



A. Behr,
Einführung in die Statistik mit R 2011

Lehrmaterial und Literatur II



J. Groß,
Grundlegende Statistik mit R, 2010,
<http://link.springer.com/book/10.1007%2F978-3-8348-9677-3>



T. Hothorn, B.S. Everitt,
A Handbook of Statistical Analyses Using R, 2014



R. Hatzinger,
R, Einführung durch angewandte Statistik, 2011



R.P. Hellbrück,
eine Einführung für Ökonomen und Sozialwissenschaftler, 2011



P. Dalgaard,
Introductory statistics with R, 2008 (e-book)



R. Schlittgen,
Einführung in die Statistik, 2008

Handbücher:



R development core team,
The R manuals,
<http://cran.r-project.org/manuals.html>



J. Groß
R Reader Arbeiten mit dem Statistikprogramm R:
<http://cran.r-project.org/doc/contrib/Grosz+Peters-R-Reader.pdf>







W. Krijnen
Applied Statistics for Bioinformatics using R, 2009:
<http://cran.r-project.org/doc/contrib/Krijnen-IntroBioInfStatistics.pdf>

Weitere Literatur unter Books related to R:





<http://r-project.org/doc/bib/R-books.html>

A. Mändle





Ziele dieses Kurses

- ▶ **Statistische Auswertungen** mittels Statistik-Software
- ▶ Einführung in  – Voraussetzungen:
 - ▶ Programmier-Verständnis
 - ▶ Grundlagen der Statistik
 - ▶ **keine** Programmier-Kenntnisse
- ▶ Der Umfang von R ist groß und ständig wachsend. In diesem Kurs lernen wir die Basisfunktionen von  kennen. Ziel:
 - mit konkreten Aufgaben einen Einstieg in  bieten,
 - grundlegende Funktionen in  kennen,
 - eigene Funktionen schreiben,
 - Handhabung von Datenstrukturen, Ein- und Ausgabe von Daten,
 - Datenanalysen ausführen (deskriptive und explorative Statistik).

Ziele dieses Kurses

- ▶ Statistische Auswertungen mittels Statistik-Software
- ▶ **Einführung** in  – Voraussetzungen:
 - ▶ Programmier-Verständnis
 - ▶ Grundlagen der Statistik
 - ▶ **keine** Programmier-Kenntnisse
- ▶ Der Umfang von R ist groß und ständig wachsend. In diesem Kurs lernen wir die Basisfunktionen von  kennen. Ziel:
 - mit konkreten Aufgaben einen Einstieg in  bieten,
 - grundlegende Funktionen in  kennen,
 - eigene Funktionen schreiben,
 - Handhabung von Datenstrukturen, Ein- und Ausgabe von Daten,
 - Datenanalysen ausführen (deskriptive und explorative Statistik).

Ziele dieses Kurses

- ▶ Statistische Auswertungen mittels Statistik-Software
- ▶ Einführung in  – Voraussetzungen:
 - ▶ Programmier-Verständnis
 - ▶ Grundlagen der Statistik
 - ▶ **keine** Programmier-Kenntnisse
- ▶ Der Umfang von R ist groß und ständig wachsend. In diesem Kurs lernen wir die **Basisfunktionen** von  kennen. Ziel:
 - mit konkreten Aufgaben einen Einstieg in  bieten,
 - grundlegende Funktionen in  kennen,
 - eigene Funktionen schreiben,
 - Handhabung von Datenstrukturen, Ein- und Ausgabe von Daten,
 - Datenanalysen ausführen (deskriptive und explorative Statistik).

Beispiel 1: Deskriptive Statistik

Die Angaben zum Geschlecht und zur Haar- und Augenfarbe von 1000 Menschen sind im Datensatz *GHA.txt* gespeichert:

	Geschlecht	Haarfarbe	Augenfarbe
1	W	black	brown
6	W	red	brown
.	.	.	.

Beispiel 1: Deskriptive Statistik

Die Angaben zum Geschlecht und zur Haar- und Augenfarbe von 1000 Menschen sind im Datensatz *GHA.txt* gespeichert:

	Geschlecht	Haarfarbe	Augenfarbe
1	W	black	brown
6	W	red	brown
.	.	.	.

Laden von Daten aus einer lokalen Datei:

```
gha <- read.table(file="C:/meinordner/GHA.txt",  
                  header=T)
```

Beispiel 1: Deskriptive Statistik

Die Angaben zum Geschlecht und zur Haar- und Augenfarbe von 1000 Menschen sind im Datensatz *GHA.txt* gespeichert:

	Geschlecht	Haarfarbe	Augenfarbe
1	W	black	brown
6	W	red	brown
.	.	.	.

Laden von Daten aus einer lokalen Datei:

```
gha <- read.table(file="C:/meinordner/GHA.txt",  
                  header=T)
```

Laden von Daten aus dem Netz:

```
gha <- read.table(url("https://pastebin.com/raw/df3x1d3J"),  
                  header=T)
```

Beispiel 1: Deskriptive Statistik

Die Angaben zum Geschlecht und zur Haar- und Augenfarbe von 1000 Menschen sind im Datensatz *GHA.txt* gespeichert:

	Geschlecht	Haarfarbe	Augenfarbe
1	W	black	brown
6	W	red	brown
.	.	.	.

Laden von Daten aus einer lokalen Datei:

```
gha <- read.table(file="C:/meinordner/GHA.txt",  
                  header=T)
```

Laden von Daten aus dem Netz:

```
gha <- read.table(url("https://pastebin.com/raw/df3x1d3J"),  
                  header=T)
```



Pfadangabe in R

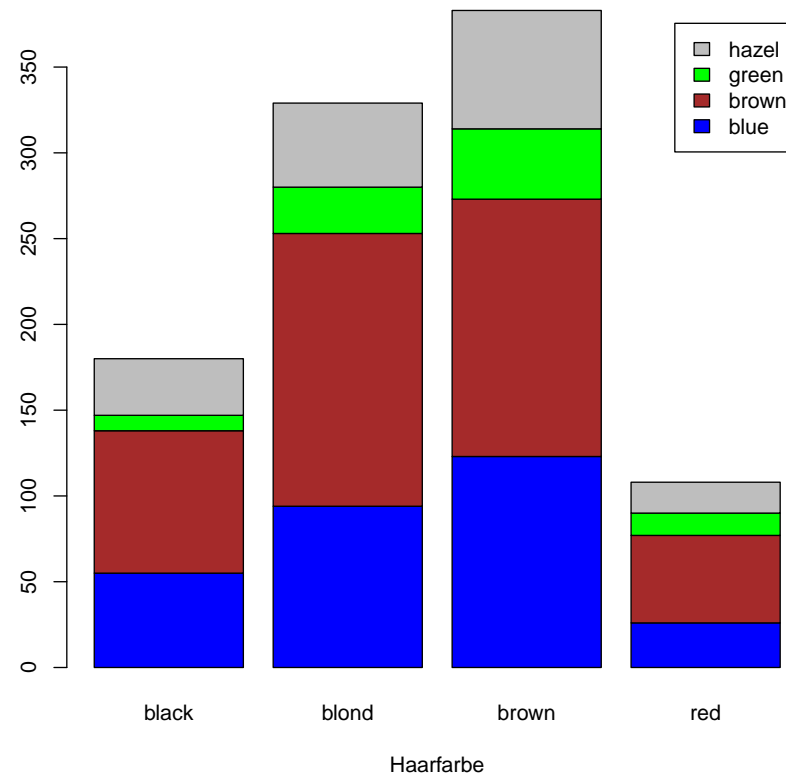
Die unter Windows üblichen Backslashes "\" müssen durch Slashes "/" oder Doppelbackslashes "\\" ersetzt werden.

Erstellen Sie eine Kontingenztafel (wie unten) zu den Daten. Stellen Sie die Daten graphisch dar.

	black	blond	brown	red
blue	55	94	123	26
brown	83	159	150	51
green	9	27	41	13
hazel	33	49	69	18

-Code:




```
table(gha$Geschlecht, gha$Augenfarbe)
table(gha$Geschlecht, gha$Haarfarbe)
table(gha$Augenfarbe, gha$Haarfarbe)
table(gha)
```

R-Befehl:

```
mytab <- table(gha$Augenfarbe,gha$Haarfarbe)
barplot(mytab, beside=F, legend=T,
        col = c("blue", "brown", "green", "gray"),
        xlab="Haarfarbe", xlim=c(0,6))
```

Hinweis

Die meisten hier angegebenen -Befehle können Sie kopieren und direkt in die -Konsole bzw. ein -Skript einfügen und verwenden.

```
rmnorm(10)
```

Beispiel 2: Simulation

Simulieren Sie das n -malige Werfen eines *fairen* Würfels und fassen Sie das Ergebnis in einer Tabelle zusammen:

Beispiel 2: Simulation

Simulieren Sie das n -malige Werfen eines *fairen* Würfels und fassen Sie das Ergebnis in einer Tabelle zusammen:

```
table(sample(1:6, 100, T)) # 100 mal würfeln
```

```
1  2  3  4  5  6  
12 19 17 16 17 19
```

Beispiel 2: Simulation

Simulieren Sie das n -malige Werfen eines *fairen* Würfels und fassen Sie das Ergebnis in einer Tabelle zusammen:

```
table(sample(1:6, 100, T)) # 100 mal würfeln
```

```
1  2  3  4  5  6  
12 19 17 16 17 19
```

```
table(sample(1:6, 1000, T)) # 1000 mal würfeln
```

```
1    2    3    4    5    6  
170 177 161 159 177 156
```

Beispiel 2: Simulation

Simulieren Sie das n -malige Werfen eines *fairen* Würfels und fassen Sie das Ergebnis in einer Tabelle zusammen:

```
table(sample(1:6,100,T)) # 100 mal würfeln
```


```
1 2 3 4 5 6  
12 19 17 16 17 19
```

```
table(sample(1:6,1000,T)) # 1000 mal würfeln
```


```
1 2 3 4 5 6  
170 177 161 159 177 156
```

```
# ein bisschen Programmieren:  
for(i in seq(100,1000,100)){  
  print(table(sample(1:6,i,T))/i)  
}
```

Beispiel 3: Statistischer Test

Simulieren Sie das 10000-malige Werfen eines *fairen* Würfels mit  und prüfen Sie anhand eines statistischen Tests, ob der Würfel *fair* ist.

Beispiel 3: Statistischer Test

Simulieren Sie das 10000-malige Werfen eines *fairen* Würfels mit  und prüfen Sie anhand eines statistischen Tests, ob der Würfel *fair* ist.

```
wuerfeln <- table(sample(1:6, 10000, T))  
chisq.test(wuerfeln, p=rep(1/6, 6))
```

Chi-squared test

data:

X-squared = 3.946, df=5, p-value = 0.5572

Beispiel 4: Statistischer Test

In einer Stadt wurden in einem Jahr 13452 Jungen und 12860 Mädchen geboren. Testen Sie die Nullhypothese H_0 , dass die Wahrscheinlichkeit für eine Jungengeburt 0.5 beträgt, zum Niveau 0.05.

Beispiel 4: Statistischer Test

In einer Stadt wurden in einem Jahr 13452 Jungen und 12860 Mädchen geboren. Testen Sie die Nullhypothese H_0 , dass die Wahrscheinlichkeit für eine Jungengeburt 0.5 beträgt, zum Niveau 0.05.

```
binom.test(x=c(13452, 12860), p=0.5, conf.level=0.95)
```

Exact binomial test

data: c(13452, 12860)

number of successes = 13452, number of trials = 26312,

p-value = 0.0002689

alternative hypothesis: true probability of success is not equal to 0.5

95 percent confidence interval:

0.5051897 0.5173070

sample estimates:

probability of success

0.5112496

Beispiel 5: Monte-Carlo-Simulation

Monte-Carlo Integration mit 

$$I = E_f[g(X)] = \int_{\mathcal{X}} g(x)f(x)dx = ?$$

wobei X Werte in \mathcal{X} annimmt und $f(x)$ eine Dichtefunktion ist.

Numerische Lösung: 1) Erzeuge eine Stichprobe (X_1, \dots, X_n) , wobei $X_i \sim f(x)$ und unabhängig sind.

2) Schätze I durch:

$$\hat{I} = \frac{\sum_{i=1}^n g(X_i)}{n}$$

Beispiel 5: Monte-Carlo-Simulation

Monte-Carlo Integration mit 

$$I = E_f[g(X)] = \int_{\mathcal{X}} g(x)f(x)dx = ?$$

wobei X Werte in \mathcal{X} annimmt und $f(x)$ eine Dichtefunktion ist.

Numerische Lösung: 1) Erzeuge eine Stichprobe (X_1, \dots, X_n) , wobei $X_i \sim f(x)$ und unabhängig sind.

2) Schätze I durch:

$$\hat{I} = \frac{\sum_{i=1}^n g(X_i)}{n}$$

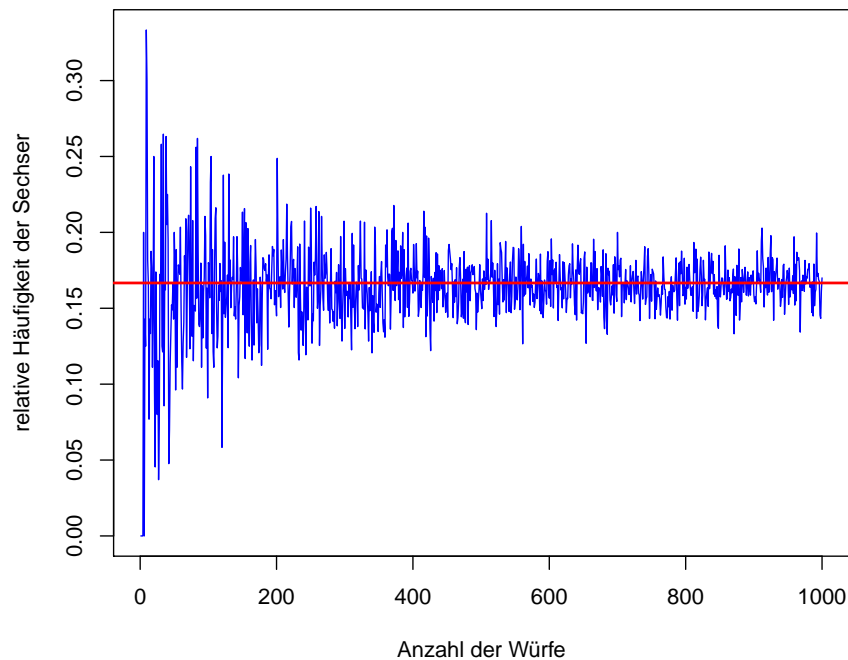
Beispiel: Berechnen Sie $\int_{-1}^1 e^{x^2} dx$

```
2*mean(exp(runif(100000,min=-1,max=1)^2))
```

Beispiel 6: GdgZ

Gesetz der großen Zahlen (GdgZ): Mit wachsendem Stichprobenumfang konvergiert der Mittelwert der Stichprobe gegen den Erwartungswert.

Relative Häufigkeit der Sechser beim j -maligem Würfeln:



Beispiel 6: GdgZ

-Skript:

```
my.fun <- function(n) {  
  x.hat <- rep(0,n)  
  for (i in 1:n) {  
    w <- sample(1:6, size=i, replace=T)  
    x.hat[i] <- sum(w==6)/i  
  }  
  x.hat  
}  
  
# und jetzt das Ergebnis grafisch darstellen:  
t <- my.fun(n=1000)  
plot(1:length(t), t, xlab="Anzahl_der_Würfe",  
      ylab="relative_Häufigkeit_der_Sechser",  
      lwd=1, col="blue", type="l")  
abline(1/6, 0, col="red", lwd=2)
```

Beispiel 7: Regressionsanalyse

```
install.packages("MASS", dependencies=T); library(MASS)
# package MASS installieren und laden
data(hills) # data hills laden
head(hills); ?hills # data hills anschauen
```

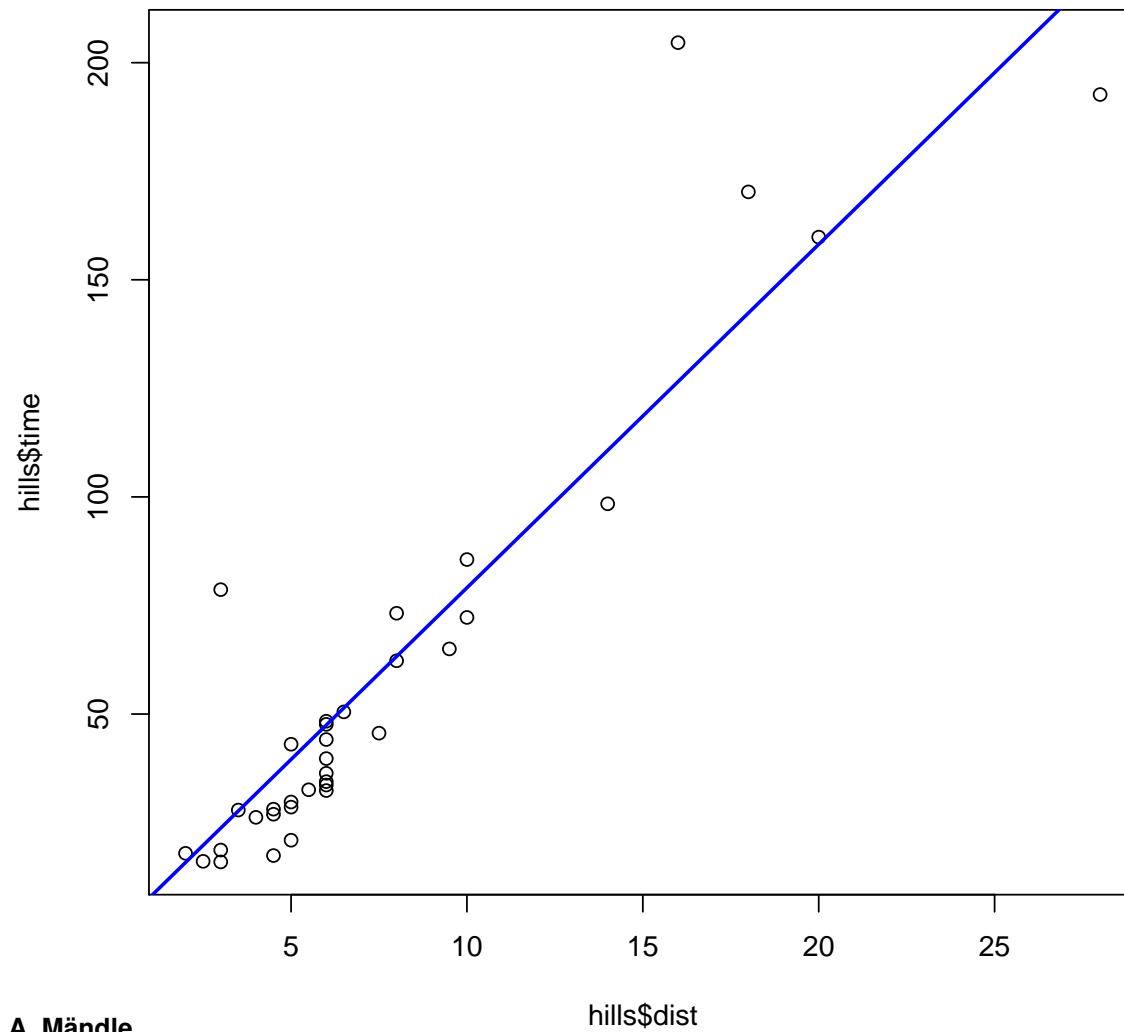
dist: distance in miles,

climb: total height gained, in feet,

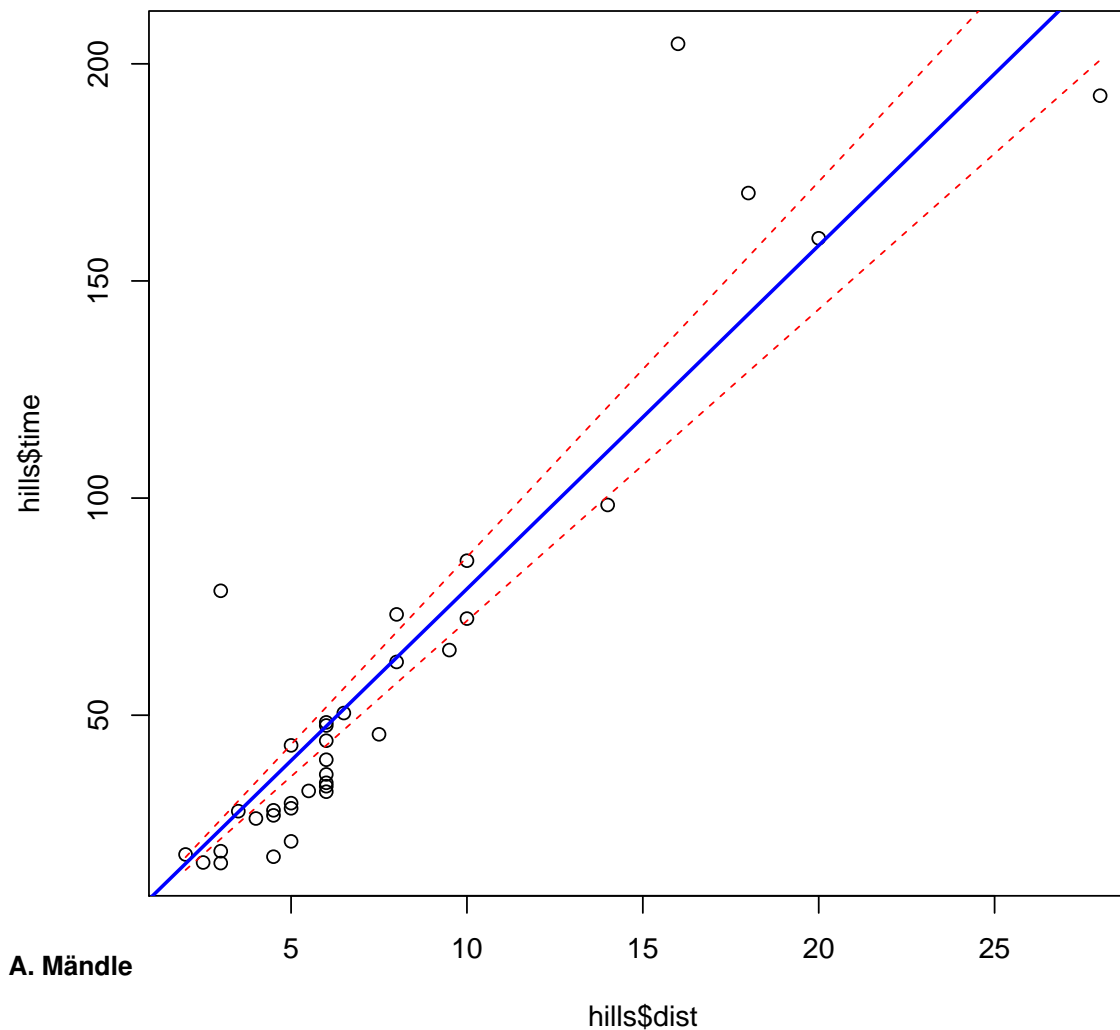
time: record time in minutes.

my.model: time~dist

```
my.model <- lm(time~dist-1, data=hills)
plot(hills$dist, hills$time)
abline(my.model, lwd = 2, col = "blue")
```











```
d <- seq(from = min(hills$dist), to = max(hills$dist), by = 0.01)
data1 <- data.frame(dist = d)
pre <- predict(my.model, newdata = data1, interval = "conf")
lines(d, pre[,2], lty = 2, col = "red")
lines(d, pre[,3], lty = 2, col = "red")
```















Gliederung I







Organisatorisches







1. Einstieg und Grundlegendes zu 	42
1.1 Hintergrund	43
1.2 Grundlegende Bedienung	56
2. Datentypen und Datenimport	92
2.1 Datentypen	93
2.2 Datenstrukturen	94
2.3 Kontrollstrukturen	148
2.4 Eingabe/Einlesen und Ausgabe von Daten	182
3. Deskriptive Statistik mit <i>R</i>	193
4. Verteilungen & Zufallszahlen in 	349
5. Schliessende Statistik: Testen und Schätzen	441







- ▶ Wurde in den 90-ern in Neuseeland entwickelt (Ihaka, Gentleman).
- ▶  ist eine nichtkommerzielle Implementierung der Programmiersprache S-Plus und deren inoffizieller *Nachfolger*.
- ▶ Warum heisst  R? Weil  von Ross und Robert entwickelt wurde. Weil R nahe an S liegt!
- ▶  ist unter `www.r-project.org` erhältlich und wird aktiv weiterentwickelt.
- ▶  ist eine (Objekt-orientierte) *Programmiersprache* inkl. eigener Programmierumgebung.
- ▶  ist eine Interpretersprache, d.h. der Programmiercode wird nicht kompiliert (sondern interpretiert).

- ▶ Wurde in den 90-ern in Neuseeland entwickelt (Ihaka, Gentleman).
- ▶  ist eine nichtkommerzielle Implementierung der Programmiersprache S-Plus und deren inoffizieller *Nachfolger*.
- ▶ Warum heisst  R? Weil  von Ross und Robert entwickelt wurde. Weil R nahe an S liegt!
- ▶  ist unter `www.r-project.org` erhältlich und wird aktiv weiterentwickelt.
- ▶  ist eine (Objekt-orientierte) *Programmiersprache* inkl. eigener Programmierumgebung.
- ▶  ist eine Interpretersprache, d.h. der Programmiercode wird nicht kompiliert (sondern interpretiert).



- ▶ Wurde in den 90-ern in Neuseeland entwickelt (Ihaka, Gentleman).
- ▶  ist eine nichtkommerzielle Implementierung der Programmiersprache S-Plus und deren inoffizieller *Nachfolger*.
- ▶ Warum heisst  R? Weil  von Ross und Robert entwickelt wurde. Weil R nahe an S liegt!
- ▶  ist unter `www.r-project.org` erhältlich und wird aktiv weiterentwickelt.
- ▶  ist eine (Objekt-orientierte) *Programmiersprache* inkl. eigener Programmierumgebung.
- ▶  ist eine Interpretersprache, d.h. der Programmiercode wird nicht kompiliert (sondern interpretiert).

- ▶ Wurde in den 90-ern in Neuseeland entwickelt (Ihaka, Gentleman).
- ▶  ist eine nichtkommerzielle Implementierung der Programmiersprache S-Plus und deren inoffizieller *Nachfolger*.
- ▶ Warum heisst  R? Weil  von Ross und Robert entwickelt wurde. Weil R nahe an S liegt!
- ▶  ist unter `www.r-project.org` erhältlich und wird aktiv weiterentwickelt.
- ▶  ist eine (Objekt-orientierte) *Programmiersprache* inkl. eigener Programmierumgebung.
- ▶  ist eine Interpretersprache, d.h. der Programmiercode wird nicht kompiliert (sondern interpretiert).



- ▶ Wurde in den 90-ern in Neuseeland entwickelt (Ihaka, Gentleman).
- ▶  ist eine nichtkommerzielle Implementierung der Programmiersprache S-Plus und deren inoffizieller *Nachfolger*.
- ▶ Warum heisst  R? Weil  von Ross und Robert entwickelt wurde. Weil R nahe an S liegt!
- ▶  ist unter `www.r-project.org` erhältlich und wird aktiv weiterentwickelt.
- ▶  ist eine (Objekt-orientierte) *Programmiersprache* inkl. eigener Programmierumgebung.
- ▶  ist eine Interpretersprache, d.h. der Programmiercode wird nicht kompiliert (sondern interpretiert).

- ▶ Wurde in den 90-ern in Neuseeland entwickelt (Ihaka, Gentleman).
- ▶  ist eine nichtkommerzielle Implementierung der Programmiersprache S-Plus und deren inoffizieller *Nachfolger*.
- ▶ Warum heisst  R? Weil  von Ross und Robert entwickelt wurde. Weil R nahe an S liegt!
- ▶  ist unter `www.r-project.org` erhältlich und wird aktiv weiterentwickelt.
- ▶  ist eine (Objekt-orientierte) *Programmiersprache* inkl. eigener Programmierumgebung.
- ▶  ist eine Interpretersprache, d.h. der Programmiercode wird nicht kompiliert (sondern interpretiert).



Vorteile und Nachteile

- ▶ Auf (fast) jedem Betriebssystem lauffähig.
- ▶ Einfache Handhabung: einfache Erzeugung von Objekten (Auswertungen) und direkter Zugriff auf alle erzeugten Objekte – keine Deklaration von Variablen notwendig...
- ▶ Flexibles Erstellen von Graphiken und interaktive Auswertung von Daten.
- ▶  enthält eine große Bibliothek von Funktionen, neue Verfahren aus der Forschung sind häufig bereits implementiert.
⇒ Es fällt schwer, einen Überblick zu finden.
- ▶  ist Kommandozeilen basiert (SPSS u.a. sind menügesteuert);
menügesteuerte Befehle erhält man mit dem Package *Rcmdr*.
- ▶ Recht langsam im Vergleich mit Python, C, Fortran, Julia,...



Vorteile und Nachteile

- ▶ Auf (fast) jedem Betriebssystem lauffähig.
- ▶ Einfache Handhabung: einfache Erzeugung von Objekten (Auswertungen) und direkter Zugriff auf alle erzeugten Objekte – keine Deklaration von Variablen notwendig...
- ▶ Flexibles Erstellen von Graphiken und interaktive Auswertung von Daten.
- ▶  enthält eine große Bibliothek von Funktionen, neue Verfahren aus der Forschung sind häufig bereits implementiert.
⇒ Es fällt schwer, einen Überblick zu finden.
- ▶  ist Kommandozeilen basiert (SPSS u.a. sind menügesteuert);
menügesteuerte Befehle erhält man mit dem Package *Rcmdr*.
- ▶ Recht langsam im Vergleich mit Python, C, Fortran, Julia,...



Vorteile und Nachteile

- ▶ Auf (fast) jedem Betriebssystem lauffähig.
- ▶ Einfache Handhabung: einfache Erzeugung von Objekten (Auswertungen) und direkter Zugriff auf alle erzeugten Objekte – keine Deklaration von Variablen notwendig...
- ▶ Flexibles Erstellen von Graphiken und interaktive Auswertung von Daten.
- ▶  enthält eine große Bibliothek von Funktionen, neue Verfahren aus der Forschung sind häufig bereits implementiert.
⇒ Es fällt schwer, einen Überblick zu finden.
- ▶  ist Kommandozeilen basiert (SPSS u.a. sind menügesteuert);
menügesteuerte Befehle erhält man mit dem Package *Rcmdr*.
- ▶ Recht langsam im Vergleich mit Python, C, Fortran, Julia,...



Vorteile und Nachteile

- ▶ Auf (fast) jedem Betriebssystem lauffähig.
- ▶ Einfache Handhabung: einfache Erzeugung von Objekten (Auswertungen) und direkter Zugriff auf alle erzeugten Objekte – keine Deklaration von Variablen notwendig...
- ▶ Flexibles Erstellen von Graphiken und interaktive Auswertung von Daten.
- ▶  enthält eine große Bibliothek von Funktionen, neue Verfahren aus der Forschung sind häufig bereits implementiert.
⇒ Es fällt schwer, einen Überblick zu finden.
- ▶  ist Kommandozeilen basiert (SPSS u.a. sind menügesteuert);
menügesteuerte Befehle erhält man mit dem Package *Rcmdr*.
- ▶ Recht langsam im Vergleich mit Python, C, Fortran, Julia,...


Vorteile und Nachteile

- ▶ Auf (fast) jedem Betriebssystem lauffähig.
- ▶ Einfache Handhabung: einfache Erzeugung von Objekten (Auswertungen) und direkter Zugriff auf alle erzeugten Objekte – keine Deklaration von Variablen notwendig...
- ▶ Flexibles Erstellen von Graphiken und interaktive Auswertung von Daten.
- ▶  enthält eine große Bibliothek von Funktionen, neue Verfahren aus der Forschung sind häufig bereits implementiert.
⇒ Es fällt schwer, einen Überblick zu finden.
- ▶  ist Kommandozeilen basiert (SPSS u.a. sind menügesteuert);
menügesteuerte Befehle erhält man mit dem Package *Rcmdr*.
- ▶ Recht langsam im Vergleich mit Python, C, Fortran, Julia,...

Vorteile und Nachteile


- ▶ Auf (fast) jedem Betriebssystem lauffähig.
- ▶ Einfache Handhabung: einfache Erzeugung von Objekten (Auswertungen) und direkter Zugriff auf alle erzeugten Objekte – keine Deklaration von Variablen notwendig...
- ▶ Flexibles Erstellen von Graphiken und interaktive Auswertung von Daten.
- ▶  enthält eine große Bibliothek von Funktionen, neue Verfahren aus der Forschung sind häufig bereits implementiert.
⇒ Es fällt schwer, einen Überblick zu finden.
- ▶  ist Kommandozeilen basiert (SPSS u.a. sind menügesteuert);
menügesteuerte Befehle erhält man mit dem Package *Rcmdr*.
- ▶ Recht langsam im Vergleich mit Python, C, Fortran, Julia,...

Was kann man mit machen?

 *is a language for data analysis and graphics.*
(R. Ihaka, R. Gentleman)

- ▶ Als Taschenrechner benutzen
- ▶ Daten laden (aus Netz oder von Festplatte)
- ▶ Graphische Darstellung der Daten
- ▶ Datenanalyse/Datenauswertung
- ▶ Simulationen durchführen

Beispiel: als Taschenrechner I

 starten \rightsquigarrow hinter dem „Prompt-Zeichen“ ($>$) kann man Befehle angeben und mittels „RETURN“-Taste ausführen:

Alles, was nach $\#$ steht, wird bei der Ausführung ignoriert!

Beispiel 1.1 (Einfache Rechenoperationen).

```
(2+5) * 3
2+5*3      # "Punkt vor Strich"
0.01*7E+4   # 7E+4 steht für 7*10^4
((4.3+2.3) * 5.4) / 78
4^2
sqrt(64)     #square root, also Wurzel ziehen
exp(2)
log(exp(2))
sqrt(-1)     # Fehler!
sqrt(-1+0i)  # ok!
1/0          # auch ok!
```


Beispiel: als Taschenrechner II

Beispiel 1.2 (Rechenoperationen und Wertzuweisung).

Anstatt die Ergebnisse direkt anzuzeigen, kann man diese mit dem Operator „<-“ in Variablen speichern:

```
x1 <- 4.3+2.3
x2 <- x1*5.4
x3 <- x2/78
x3
ls()           # listet die existierenden Objekte auf
rm(x1,x2,x3)  # Objekte entfernen
y1 <- c(2,4,6,8)
y2 <- y1*5
y2
y3 <- mean(y2); y4 <- sd(y2)
y3; y4
```

Bemerkungen I

- Variablennamen sollen mit einem Buchstaben oder einem Punkt beginnen, gefolgt von weiteren Buchstaben, Ziffern, Punkt oder dem Sonderzeichen `_`:

`Myvar, myvar, myvar1, myvar.y, .myvar`

- R unterscheidet zwischen Groß- und Kleinschreibung:

`x <- 5; X <- 6`

- Einige Namen sind von  reserviert (daher nicht erlaubt)

`for, in, while, repeat, ...`

- Wird einer Variablen ein neuer Wert zugewiesen, wird der alte Wert überschrieben:

`x <- 5`

`x <- 6*12`

`x`

72

Bemerkungen I

- Variablennamen sollen mit einem Buchstaben oder einem Punkt beginnen, gefolgt von weiteren Buchstaben, Ziffern, Punkt oder dem Sonderzeichen `_`:

`Myvar, myvar, myvar1, myvar.y, .myvar`

- R unterscheidet zwischen Groß- und Kleinschreibung:

`x <- 5; X <- 6`

- Einige Namen sind von  reserviert (daher nicht erlaubt)
`for, in, while, repeat, ...`

- Wird einer Variablen ein neuer Wert zugewiesen, wird der alte Wert überschrieben:

```
x <- 5
x <- 6*12
x
```

72

Bemerkungen I

- Variablennamen sollen mit einem Buchstaben oder einem Punkt beginnen, gefolgt von weiteren Buchstaben, Ziffern, Punkt oder dem Sonderzeichen `_`:

`Myvar, myvar, myvar1, myvar.y, .myvar`

- R unterscheidet zwischen Groß- und Kleinschreibung:

`x <- 5; X <- 6`

- Einige Namen sind von  reserviert (daher nicht erlaubt)
`for, in, while, repeat, ...`

- Wird einer Variablen ein neuer Wert zugewiesen, wird der alte Wert überschrieben:

```
x <- 5
x <- 6*12
x
```

72

Bemerkungen I

- Variablennamen sollen mit einem Buchstaben oder einem Punkt beginnen, gefolgt von weiteren Buchstaben, Ziffern, Punkt oder dem Sonderzeichen _:

`Myvar, myvar, myvar1, myvar.y, .myvar`

- R unterscheidet zwischen Groß- und Kleinschreibung:

`x <- 5; X <- 6`

- Einige Namen sind von  reserviert (daher nicht erlaubt)

`for, in, while, repeat, ...`

- Wird einer Variablen ein neuer Wert zugewiesen, wird der alte Wert überschrieben:

`x <- 5`

`x <- 6*12`

`x`

72

Bemerkungen II

- Mehrere Befehle ausführen (mit Semikolon trennen):

```
x <- 4;    y <- 6
```

- Wenn ein Befehl unvollständig ist, gibt  ein $+$ aus.
Man kann dann hinter dem $+$ den Befehl vervollständigen:

```
x <- 6 * (7 - 3  
+
```

Jetzt den Befehl vervollständigen:

```
+    )
```

24


- Man kann (auch) in der Konsole einen Befehl in mehrere Zeilen schreiben:

```
plot(x, y, xlim = c(0, 10), ylim = c(0, 0.5),  
+     xlab = "X", ylab = "Density", main = "Mein_Bild", ...)
```

Bemerkungen II

- Mehrere Befehle ausführen (mit Semikolon trennen):

```
x <- 4;    y <- 6
```

- Wenn ein Befehl unvollständig ist, gibt  ein **+** aus.
Man kann dann hinter dem **+** den Befehl vervollständigen:

```
x <- 6 * (7 - 3  
+
```

Jetzt den Befehl vervollständigen:

```
+    )
```

24


- Man kann (auch) in der Konsole einen Befehl in mehrere Zeilen schreiben:

```
plot(x, y, xlim = c(0, 10), ylim = c(0, 0.5),  
+     xlab = "X", ylab = "Density", main = "Mein_Bild", ...)
```

Bemerkungen II

- Mehrere Befehle ausführen (mit Semikolon trennen):

```
x <- 4;      y <- 6
```

- Wenn ein Befehl unvollständig ist, gibt  ein **+** aus.
Man kann dann hinter dem **+** den Befehl vervollständigen:

```
x <- 6 * (7 - 3  
+
```

Jetzt den Befehl vervollständigen:

```
+      )
```

24

- Man kann (auch) in der Konsole einen Befehl in mehrere Zeilen schreiben:

```
plot(x, y, xlim = c(0, 10), ylim = c(0, 0.5),  
+     xlab = "X", ylab = "Density", main = "Mein_Bild", ...)
```


Bemerkungen III

- Die Cursor-Tasten (Pfeil nach oben/unten) dienen zum Durchblättern der Befehlshistorie.
- Zuweisung ausführen und Ergebnis gleich ausgeben:

```
(x <- 5)
```

5

-  kennt komplexe Zahlen.

```
(2-3i) * (1+1i)    # versuche: (2-3i) * (1+i)
```

5-1i

- aber Achtung:

```
sqrt(-1)
```

Warning message: In sqrt(-1) : NaNs produced

Bemerkungen III

- Die Cursor-Tasten (Pfeil nach oben/unten) dienen zum Durchblättern der Befehlshistorie.
- Zuweisung ausführen und Ergebnis gleich ausgeben:

```
(x <- 5)
```

5

-  kennt komplexe Zahlen.

```
(2-3i) * (1+1i)    # versuche: (2-3i) * (1+i)
```

5-1i

- aber Achtung:

```
sqrt(-1)
```

Warning message: In sqrt(-1) : NaNs produced

Bemerkungen III

- Die Cursor-Tasten (Pfeil nach oben/unten) dienen zum Durchblättern der Befehlshistorie.
- Zuweisung ausführen und Ergebnis gleich ausgeben:

```
(x <- 5)
```

5

- R kennt komplexe Zahlen.

```
(2-3i) * (1+1i)      # versuche: (2-3i) * (1+i)
```

5-1i

- aber Achtung:

```
sqrt(-1)
```

Warning message: In sqrt(-1) : NaNs produced

Bemerkungen III

- Die Cursor-Tasten (Pfeil nach oben/unten) dienen zum Durchblättern der Befehlshistorie.
- Zuweisung ausführen und Ergebnis gleich ausgeben:

```
(x <- 5)
```

5

-  kennt komplexe Zahlen.

```
(2-3i) * (1+1i)      # versuche: (2-3i) * (1+i)
```

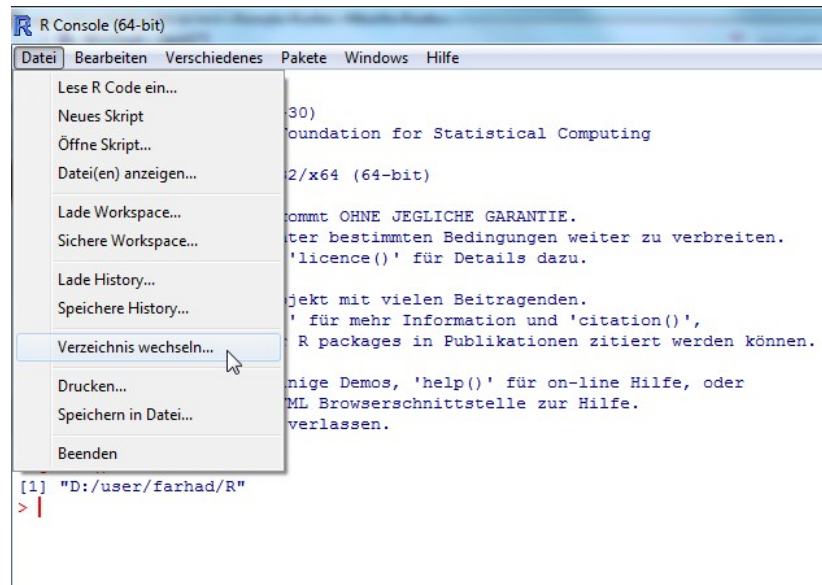
5-1i

- aber Achtung:

```
sqrt(-1)
```

Warning message: In sqrt(-1) : NaNs produced

Arbeits-Verzeichnis I



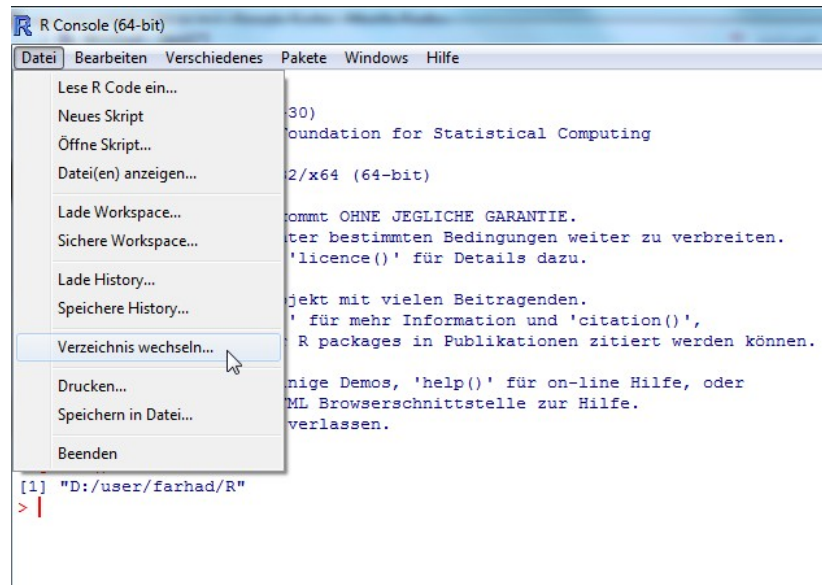
Pfade werden absolut oder relativ zum Arbeitsverzeichnis angegeben. Wenn kein Pfad spezifiziert wird, werden alle Dateien in diesem Verzeichnis gesucht und gespeichert.

Entweder über die Menüleiste wechseln:

Datei \rightsquigarrow *Verzeichnis wechseln* in der *RGui* bzw.

Session \rightsquigarrow *Set Working Directory* in *R-Studio*

Arbeits-Verzeichnis I



Pfade werden absolut oder relativ zum Arbeitsverzeichnis angegeben. Wenn kein Pfad spezifiziert wird, werden alle Dateien in diesem Verzeichnis gesucht und gespeichert.

Entweder über die Menüleiste wechseln:

Datei \rightsquigarrow *Verzeichnis wechseln* in der *RGui* bzw.

Session \rightsquigarrow *Set Working Directory* in *R-Studio*

Oder mittels Programmcode:

```
getwd()
```


```
[1] "C:/..."
```

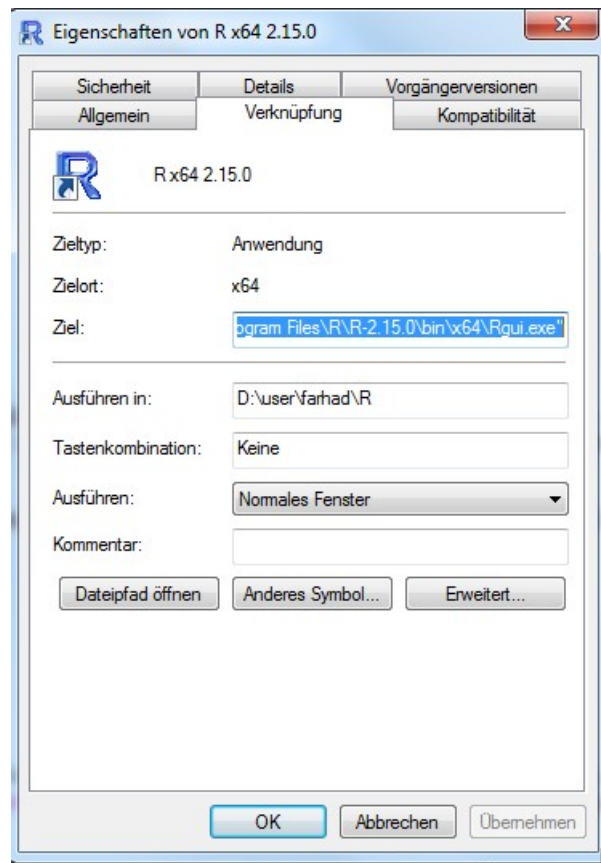
```
setwd("C:/Users/maendle/OneDrive/Farhad/WS_17_18/")
```

```
getwd()
```

```
[1] "C:/Users/maendle/OneDrive/Farhad/WS_17_18/"
```

Arbeits-Verzeichnis II

Die Änderung bezieht sich nur auf die aktuelle -Sitzung. Um automatisch mit einem frei gewählten Arbeitsverzeichnis zu starten, kann man z.B. über die Einstellung der Verknüpfung auf dem Desktop anpassen:





Auf die *RGui*-Verknüpfung mit der rechten Maustaste klicken. Dann unter *Eigenschaften* \rightsquigarrow *Verknüpfung* \rightsquigarrow *Ausführen in* ein neues Verzeichnis angeben; z.B.

D:\user\username\R

In *R-Studio* kann das Verzeichnis unter *Tools* \rightsquigarrow *Global Options* im Reiter *General* festgelegt werden.

Pakete einbinden I

- ▶  ist in Paketen organisiert.
- ▶ Ein Paket enthält Codes, Hilfeseiten, Datensätze, Beispiele usw. zu einem bestimmten Themengebiet.
- ▶ Mit der Installation von  erhält man eine Grundausstattung an wichtigen Paketen.
- ▶ Weitere Pakete können direkt über die R-Konsole **installiert** und **geladen** werden. Eine enorme Anzahl von Zusatzpaketen steht hierfür im offiziellen CRAN-Repository zur Verfügung.
- ▶ Andere Quellen für Pakete sind z.B. Bioconductor (Bio-Statistik), Omegahat (Statistical Computing) und GitHub

Pakete einbinden II

Installation von Zusatzpaketen aus dem offiziellen Repository mittels der Funktion `install.packages()`, z.B.:

► `install.packages("fun", dependencies=TRUE)`

Laden (Aktivieren) von installierten Paketen

► `library("fun")` oder `library(fun)`

Jetzt kann man die Funktionen aus dem Paket verwenden:

```
library(help="fun") # Welche Funktionen enthält das Paket?
?mine_sweeper      # Hilfe zu einer Funktion aus dem Paket
if (.Platform$OS.type == "windows") {
  x11()
} else x11(type = "Xlib")
mine_sweeper()
```

Freigeben geladener Pakete: `detach(package:fun)`

Löschen eines Pakets: `remove.packages("fun")`

Auflisten aller bereits installierten Pakete: `library()`

Pakete einbinden II

Installation von Zusatzpaketen aus dem offiziellen Repository mittels der Funktion `install.packages()`, z.B.:

► `install.packages("fun", dependencies=TRUE)`

Laden (Aktivieren) von installierten Paketen

► `library("fun")` oder `library(fun)`

Jetzt kann man die Funktionen aus dem Paket verwenden:


```
library(help="fun") # Welche Funktionen enthält das Paket?
?mine_sweeper      # Hilfe zu einer Funktion aus dem Paket
if (.Platform$OS.type == "windows") {
  x11()
} else x11(type = "Xlib")
mine_sweeper()
```

Freigeben geladener Pakete: `detach(package:fun)`

Löschen eines Pakets: `remove.packages("fun")`

Auflisten aller bereits installierten Pakete: `library()`

Hilfesystem I

- ▶ `help.start()`
öffnet die (Web-)Links zu den -Handbüchern.
- ▶ Hilfe zu bestimmten Funktionen aufrufen:
`?sqrt`
`help(sqrt)`
`help("sqrt")`
- ▶ der Befehl
`apropos("mean")`
liefert eine Liste der Funktionsnamen, welche den Ausdruck *mean* enthalten.

Bemerkung: Mit `?`, `help()`, `apropos()` wird nur Hilfe zu den Funktionen in den geladenen Paketen angezeigt.

Hilfesystem II

- ▶ `help(glm, try.all.packages=T)` sucht in allen installierten Paketen und liefert die Namen dieser Pakete. Die Hilfeseite zu einer Funktion in einem noch nicht geladenen Paket kann dann mit der Option *package* angezeigt werden.
`help(glm, package="stats")`
- ▶ `help.search("truncated")` oder `??truncated` sucht nach einem bestimmten Wort (hier *truncated*) in der Online-Hilfe.
- ▶ `RSiteSearch("truncated")` durchsucht Hilfeseiten, Vignetten und Task Views nach Stichwort "truncated".
- ▶ `help.request()` sendet Nachricht an R-help – und bietet einen Überblick über Hilfe-Quellen.

Skript erstellen

Um ein Programm zu schreiben, speichert man eine Ansammlung von R-Befehlen in einer Textdatei (*R-Skript*) mit der Dateierweiterung ".R".

In der **RGui** kann man einen Texteditor über die R-Menuleiste öffnen:
Datei ~> *Neues Skript*

Dort kann man R-Befehle eingeben, speichern und zeilenweise ausführen lassen:

Bearbeiten ~> *Ausführung Zeile oder Auswahl*
bzw.

Bearbeiten ~> *Alles ausführen*

In **R-Studio** geht das über *File* ~> *New File* ~> *R Script* und den *Run*-Button über dem Texteditor.

Skript erstellen

Um ein Programm zu schreiben, speichert man eine Ansammlung von R-Befehlen in einer Textdatei (*R-Skript*) mit der Dateiendung ".R".

In der **RGui** kann man einen Texteditor über die R-Menuleiste öffnen:
Datei ~> *Neues Skript*

Dort kann man R-Befehle eingeben, speichern und zeilenweise ausführen lassen:

Bearbeiten ~> *Ausführung Zeile oder Auswahl*
bzw.

Bearbeiten ~> *Alles ausführen*

In **R-Studio** geht das über *File* ~> *New File* ~> *R Script* und den *Run*-Button über dem Texteditor.



R-editor (R-Skript)

- Ihre Lösungen/Auswertungen schreiben Sie in einem R-Skript.
- Erläutern Sie den Code mit passenden Kommentaren (hinter #).

```
#####  
# Blatt 02  
# 01.01.1900  
# Farhad Arzideh und ...  
#####  
# Hausaufgabe 1  
#####  
x<-c(2,5,7)  
y<-1:3  
x/y  
#####  
#####  
# Hausaufgabe 2  
#####  
x <- seq(-4,4,0.05)  
x-1  
#####
```

```
#####  
# Blatt 02  
# 01.01.1900  
# Farhad Arzideh und ...  
#####  
# Hausaufgabe 1  
#####  
x<-c(2,5,7)  
y<-1:3  
x/y  
#####  
# farhad:  
# x/y ist falsch. Richtig ist:  
# x*y|  
#####  
# Hausaufgabe 2  
#####
```

Mathematische Operatoren & Funktionen

arithmetische Operatoren	<code>+</code> , <code>-</code> , <code>*</code> , <code>/</code> , <code>**</code> oder <code>^</code>
Ganzzahlige Division, Modulo	<code>%/</code> , <code>%%</code>
Matrix-Multiplikation	<code>%*%</code>
Extremwerte, Betrag	<code>min()</code> , <code>max()</code> , <code>abs()</code>
Quadratwurzel	<code>sqrt()</code>
Runden (Ab-, Auf-)	<code>round()</code> , <code>floor()</code> , <code>ceiling()</code>
Summe, Produkt	<code>sum()</code> , <code>prod()</code>
Logarithmen	<code>log()</code> , <code>log10()</code>
Exponential	<code>exp()</code>
Trigonometrische Funktionen	<code>sin()</code> , <code>cos()</code> , <code>tan()</code>
π	<code>pi</code>
∞ , $-\infty$	<code>Inf</code> , <code>-Inf</code>
nicht definiert	<code>NaN</code>
fehlende Werte	<code>NA</code>
Leere Menge	<code>NULL</code>

Logische Operatoren & Funktionen

gleich, ungleich

`==, !=`

größer, größer gleich

`>, >=`

kleiner, kleiner gleich

`<, <=`

nicht (Negation)

`!`

und, oder

`& (&&), | (||)`

entweder oder (ausschließend)

`xor()`

wahr, falsch

`TRUE, FALSE`

Beispiele

```
5 != 2+2
```

```
TRUE
```

```
TRUE & FALSE
```

```
FALSE
```

```
TRUE | FALSE
```

```
TRUE
```

```
(4 >= 3 | 5 == 6) & (4 != 4)
```

```
FALSE
```

```
FALSE || (2 != 3)
```

```
TRUE
```

```
help("||"), help("+")
```

```
7 %/% 2
```

```
3
```

```
18 %% 3
```

```
0
```

```
Inf + Inf + 2
```

```
Inf
```

```
Inf - Inf + 2
```

```
NaN
```

```
5/0
```

```
Inf
```



Vorsicht: Unterschied zwischen & und && (bzw. | und ||)

|, &: prüfen Wahrheitswerte zweier Vektoren komponentenweise.
&&, ||: prüfen nur das jeweils erste (!) Element zweier Vektoren.

Übung 1.1

Evaluieren Sie die folgenden Aussagen zuerst ohne Anwendung einer Software und dann mit R:

```
x <- 1
(x > 2) & (x < 5)
(x > 2) && (x < 5)
(x > 2) | (x < 5)
(x > 2) || (x < 5)
```

```
x <- 1:7
(x > 2) & (x < 5)
(x > 2) && (x < 5)
(x > 2) | (x < 5)
(x > 2) || (x < 5)
```

```
c(FALSE, TRUE, TRUE) && c(TRUE, TRUE, TRUE)
c(TRUE, FALSE, TRUE) && c(TRUE, TRUE, TRUE)
```

Vektoren

Vektoren sind in **R** elementar; Skalare gibt es nicht im eigentlichen Sinne, sondern sie werden durch Vektoren der Länge 1 dargestellt. Mit der Funktion `c()` (für *combine* bzw. *concatenate*) definiert man einen Vektor. Beispiele:

```
x <- c(1,2,3,4); x ..... 1 2 3 4
y <- c(4,3,2,1); y ..... 4 3 2 1
z <- c(1:4); z ..... 1 2 3 4
w <- c(4:1); w ..... 4 3 2 1
w==y ..... TRUE TRUE TRUE TRUE
u <- seq(1,4); u ..... 1 2 3 4
(x==z) & (z==u) ..... TRUE TRUE TRUE TRUE
a <- c("Einführung","in R"); a "Einführung" "in R"
b <- c(x,u); b ..... 1 2 3 4 1 2 3 4
e <- c(a,u); e ..... "Einführung" "in R" "1" "2" "3" "4"
```

Funktionen verwenden

Beispiel: `seq()`

```
seq(from=1, to=1, by=((to-from)/(length.out-1)),  
length.out=NULL, ...)
```

from, to, ... sind die Argumente (Optionen) der Funktion

```
seq(from=2, to=6)  
seq(2, 6)  
seq(from=2, to=6, by=2)  
seq(from=2, to=6, by=3)  
seq(6, 2)  
seq(6, 2, 1) # Fehlermeldung  
seq(to=10, from=3, by=2)  
seq(by=2, to=10, from=3)  
seq(4, 5, length=10)  
seq(4, 5, len=10)  
# from:to ist äquivalent zu seq(from, to)  
# (sofern Parameter keine Faktoren sind)  
1:6  
10:2
```

Funktionen definieren

Eigene Funktionen kann man wie im folgenden Beispiel definieren:

```
meine.fun <- function(x1 ,x2) (x1 + x2)/2 # oder
meine.fun <- function(x1 ,x2) {
  (x1 + x2)/2
} # oder
meine.fun <- function(x1 ,x2) {
  y <- (x1 + x2)/2
  y
} # oder
meine.fun <- function(x1 ,x2) {
  y <- (x1 + x2)/2
  return(y)
}
```

#Aufruf der Funktion:

```
meine.fun(100, 200) # oder
meine.fun(x1 = 100, x2 = 200) # oder
meine.fun(x2 = 200, x1 = 100)
```


Übung 1.2

1) Die Funktion `mean()` berechnet den Mittelwert eines Vektors. Schreiben Sie **Ihre eigene** Funktion um den Mittelwert aus beliebig vielen Zahlen, erfasst in einem Vektor, zu berechnen.

Hinweis: benutzen Sie die Funktionen: `sum()` und `length()`.


2) Schreiben Sie **Ihre eigene** Funktion um die Intervall-Länge (`range`) eines Vektors zu berechnen.

R Beenden

- Der Befehl `q()` beendet die -Sitzung.
- Sichern von *History* und *Workspace* (beides vermeiden!):
Die *History* kann man in der Datei `.Rhistory` im aktuellen Arbeitsverzeichnis speichern (=Abspeichern zuletzt ausgeführter Befehle).
Den *Workspace* kann man in der Datei `.RData` im aktuellen Arbeitsverzeichnis speichern (=Abspeichern aller aktuellen Objekte).



Beenden einer R-Sitzung

Beim Beenden einer -Sitzung wird nachgefragt, ob der aktuelle *workspace* (d.h. alle aktuellen Objekte) gespeichert werden soll. I.d.R. beantworten Sie diese Frage mit **nein**.

R Beenden


- Der Befehl `q()` beendet die R-Sitzung.
- Sichern von *History* und *Workspace* (beides vermeiden!):
Die *History* kann man in der Datei `.Rhistory` im aktuellen Arbeitsverzeichnis speichern (=Abspeichern zuletzt ausgeführter Befehle).
Den *Workspace* kann man in der Datei `.RData` im aktuellen Arbeitsverzeichnis speichern (=Abspeichern aller aktuellen Objekte).



Beenden einer R-Sitzung


Beim Beenden einer R-Sitzung wird nachgefragt, ob der aktuelle *workspace* (d.h. alle aktuellen Objekte) gespeichert werden soll. I.d.R. beantworten Sie diese Frage mit **nein**.

R Beenden


- Der Befehl `q()` beendet die -Sitzung.
- Sichern von *History* und *Workspace* (beides vermeiden!):
Die *History* kann man in der Datei `.Rhistory` im aktuellen Arbeitsverzeichnis speichern (=Abspeichern zuletzt ausgeführter Befehle).
Den *Workspace* kann man in der Datei `.RData` im aktuellen Arbeitsverzeichnis speichern (=Abspeichern aller aktuellen Objekte).



Beenden einer R-Sitzung


Beim Beenden einer -Sitzung wird nachgefragt, ob der aktuelle *workspace* (d.h. alle aktuellen Objekte) gespeichert werden soll. I.d.R. beantworten Sie diese Frage mit **nein**.

R Beenden

- Der Befehl `q()` beendet die -Sitzung.
- Sichern von *History* und *Workspace* (beides **vermeiden!**):
Die *History* kann man in der Datei `.Rhistory` im aktuellen Arbeitsverzeichnis speichern (=Abspeichern zuletzt ausgeführter Befehle).
Den *Workspace* kann man in der Datei `.RData` im aktuellen Arbeitsverzeichnis speichern (=Abspeichern aller aktuellen Objekte).





Beenden einer R-Sitzung

Beim Beenden einer -Sitzung wird nachgefragt, ob der aktuelle *workspace* (d.h. alle aktuellen Objekte) gespeichert werden soll. I.d.R. beantworten Sie diese Frage mit **nein**.

Gliederung I

Organisatorisches

1. Einstieg und Grundlegendes zu 	42
1.1 Hintergrund	43
1.2 Grundlegende Bedienung	56
2. Datentypen und Datenimport	92
2.1 Datentypen	93
2.2 Datenstrukturen	94
2.3 Kontrollstrukturen	148
2.4 Eingabe/Einlesen und Ausgabe von Daten	182
3. Deskriptive Statistik mit <i>R</i>	193
4. Verteilungen & Zufallszahlen in 	349
5. Schliessende Statistik: Testen und Schätzen	441

Einfache Datentypen

<i>Datentyp</i>	Beschreibung	Beispiel
<i>NULL</i>	leere Menge	NULL
<i>logical</i>	logische Werte	TRUE
<i>character</i>	Zeichenketten	"text"
<i>factor</i>	kategorielle oder ordinale Merkmale	female
<i>integer</i>	ganze Zahlen	-3
<i>numeric</i>	ganze oder reelle Zahlen	-2e-6
<i>complex</i>	komplexe Zahlen	2-1i

Vektoren eines Datentypes können mit Funktionen erzeugt werden, die meist den Namen des Datentyps tragen:

```
as.complex(), as.numeric(), as.integer(),  
as.factor(), as.character()
```

Überprüfen des Typs mittels der Funktionen:

```
is.complex(), is.numeric(), is.integer(),  
is.factor(), is.character()
```

`class()` ermittelt den Datentyp des übergebenen Objekts.

Datenstrukturen und Behandlung

- 1 Vektoren
- 2 Matrizen
- 3 Arrays
- 4 Listen
- 5 Data frames (Datensätze)
- 6 Logische Operationen zur Indizierung von Daten

Vektoren: Rechnen mit Vektoren

```
numeric(5) ..... 0 0 0 0 0
c(1,2,3,4)**2 ..... 1 4 9 16
c(1,2,3,4)-1 ..... 0 1 2 3
c(1,2,3,4)*c(1,-1,0,2) .... 1 -2 0 8
c(1,2,3,4)*c(3,-1) ..... 3 -2 9 -4
c(1,2,3,4)*c(1,2,3) ..... 1 4 9 4 # Warnmeldung: ...
x<-c(10,11,15); length(x) 3
sum(x) 36
```

Vektoren erzeugen mit *seq()* und *rep()*

<code>seq(5, 7)</code>	5 6 7
<code>seq(10, 15, by=2)</code>	10 12 14
<code>seq(1, 10, len=5)</code>	1 3.25 5.5 7.75 10
<code>rep(4, 3)</code>	4 4 4
<code>rep(c(1, 0), 3)</code>	1 0 1 0 1 0
<code>rep(c("M", "W"), 3)</code>	"M" "W" "M" "W" "M" "W"
<code>c(rep(2, 3), rep("M", 3))</code>	?
<code>rep(c(1, 0), times=3)</code>	1 0 1 0 1 0
<code>rep(c(1, 0), each=3)</code>	1 1 1 0 0 0
<code>rep(c(1, 0), times=c(2, 3))</code>	1 1 0 0 0
<code>rep(c(1, 0), each=2, times=3)</code> ..	?

Vektoren: *Indizierung*

```
x <- c(12, 18, 23, 5, 9); x[2] .... 18
x[3:4] ..... 23 5
x[c(1, 5)] ..... 12 9
x[c(1, length(x))] ..... 12 9
x[-c(1, 5)] ..... 18 23 5
x[3] <- 99; x ..... 12 18 99 5 9
x[3]*log(2) ..... 68.62157
```

Elemente eines Vektors benennen:

```
x <- c(A=5, B=4); x
```

A B
5 4

```
x["A"]
```

A
5

```
x[1]
```

A
5

```
names(x) <- c("1.Element",  
              "2.Element"); x
```

1.Element 2.Element
5 4

Vektoren: *Datentyp ändern*

```
x1 <- c(12,18,23,5,9) .....  
is.numeric(x1) ..... TRUE  
x2 <- c(rep("M",2),rep("W",2))  
is.numeric(x2) ..... FALSE  
is.factor(x2) ..... FALSE  
is.character(x2) ..... TRUE  
x3 <- as.factor(x2) .....  
x3 ..... M M W W  
is.factor(x3) ..... TRUE
```

Matrizen *definieren*

```
matrix(data = NA, nrow = 1, ncol = 1,  
        byrow = FALSE, dimnames = NULL)
```

Führen Sie die folg. Befehle aus:

```
X1 <- matrix(1:4, nrow=2, ncol=2);    X1  
X2 <- matrix(1:4, nrow=3);    X2  
X3 <- matrix(c(1, 0, 1, 0), nrow=2, byrow=TRUE);    X3  
X4 <- matrix(c(1, 0, 1, 0), nrow=2, byrow=FALSE);    X4  
X5 <- matrix(c(1, 2, 3, 10, 20, 30),  
             nrow = 3, ncol=2, byrow=FALSE,  
             dimnames = list(c("row1", "row2", "row3"),  
                              c("col1", "col2"))); X5
```

Man kann die Spalten und Zeilen einer Matrix später (um-)benennen:

```
X5 <- matrix(c(1,2,3, 10,20,30), nrow = 3, ncol=2)
colnames(X5) <- c("col1", "col2")
rownames(X5) <- c("row1", "row2", "row3")
# oder in einem Schritt mit:
dimnames(X5) <- list(c("row1", "row2", "row3"),
                    c("col1", "col2"))
```

Matrizen: *wichtige Befehle und Operatoren*

<code>dim(X)</code>	<code>a*X</code> (Multiplikation mit Skalar)
<code>nrow(X)</code>	<code>X1%*%X2</code> (Matrizenmultiplikation)
<code>ncol(X)</code>	<code>t(X)</code> (transponieren)
<code>dimnames(X)</code>	<code>solve(X)</code> (Inversmatrix)
<code>rownames(X)</code>	<code>eigen(X)</code> (Eigenwert)
<code>colnames(X)</code>	<code>det(X)</code>
<code>diag(X)</code>	<code>cbind(X1,X2)</code>
	<code>rbind(X1,X2)</code>

Rufen Sie auch ggf. die Hilfe-Funktionen zu den o.g. Befehlen auf!

Matrizen: *Indizierung*

```
X <- matrix(c(1,2,3, 10,20,30, 100,200,300),nrow=3)
```

$$X = \begin{pmatrix} 1 & 10 & 100 \\ 2 & 20 & 200 \\ 3 & 30 & 300 \end{pmatrix}$$

```
X[2,2]      20
```

```
X[2,]      2  20  200
```

```
X[,3]     100  200  300
```

```
X[1:2,1:2]   $\begin{pmatrix} 1 & 10 \\ 2 & 20 \end{pmatrix}$ 
```

```
X[, -c(1,3)]  10  20  30
```

```
X[, -c(1,3), drop=FALSE]  $\begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix}$ 
```

Einzeilige, bzw. einspaltige Matrizen werden automatisch in Vektoren konvertiert, sofern nicht das Attribut `drop=FALSE` gesetzt wurde.

Matrizen: *Indizierung*

Analog zu Vektoren kann man bei der Indizierung die *Zeilen-* bzw. *Spaltennamen* der Matrix nutzen (sofern vorhanden). Bsp.:

```
(X5 <- matrix(c(1,2,3, 10,20,30), nrow = 3, ncol=2,  
  dimnames = list(c("row1", "row2", "row3"),  
    c("col1", "col2"))))
```

$$X5 = \begin{pmatrix} 1 & 10 \\ 2 & 20 \\ 3 & 30 \end{pmatrix}$$

```
X5["row1",]  
1  10
```

```
X5[, "col2"]  
10  20  30
```

```
X5["row1", "col2"]  
  
10
```

Teilmenge von Vektoren und Matrizen

```
x <- c(5, 12, 26, -2.1); x  
5.0 12.0 26.0 -2.1
```

```
X <- matrix(c(1, 2, 3, 10, 20, 30), nrow = 3); X  

$$X = \begin{pmatrix} 1 & 10 \\ 2 & 20 \\ 3 & 30 \end{pmatrix}$$

```

```
x[x > 0]  
5 12 26
```

```
x[x > 0 & x < 20]  
5 12
```

```
X[X[, 2] > 10,]  

$$\begin{pmatrix} 2 & 20 \\ 3 & 30 \end{pmatrix}$$

```

```
X[X[, 1] < 2,]  
1 10
```


Arrays

Arrays können beliebig viele Dimensionen besitzen:

```
XX <- array(1:30, dim=c(2, 5, 3))
```

```
, , 1
```

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

```
, , 2
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	11	13	15	17	19
[2,]	12	14	16	18	20

```
, , 3
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	21	23	25	27	29
[2,]	22	24	26	28	30

Übung 2.1

Definieren Sie das *Array* `XX<- array(1:30, dim=c(2,5,3))` und machen Sie sich vertraut mit dem Umgang mit der Indizierung von *Arrays*.

`XX[, , 2], XX[, 3 , 2] , XX[1, 3 , 2]` usw.

Listen

Listen besitzen eine sehr flexible Struktur: Die Elemente einer Liste können unterschiedliche Datentypen besitzen; z.B. verschieden lange Vektoren oder/und Matrizen unterschiedlichen Typs enthalten, oder selbst wieder Element einer Liste sein.

Beispiel:

```
x <- c(1, 2)
X <- matrix(c(1, 2, 3, 10, 20, 30), nrow = 3)
XX <- array(1:30, dim=c(2, 5, 3))
Y <- list(x, X, XX)
```

```
[[1]]
```

```
[1] 1 2
```

```
[[2]]
```

	[, 1]	[, 2]
[1,]	1	10
[2,]	2	20
[3,]	3	30

[[3]]
 , , 1

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

, , 2

	[, 1]	[, 2]	[, 3]	[, 4]	[, 5]
[1,]	11	13	15	17	19
[2,]	12	14	16	18	20

, , 3

	[, 1]	[, 2]	[, 3]	[, 4]	[, 5]
[1,]	21	23	25	27	29
[2,]	22	24	26	28	30

Listen: *Indizierung*

```
Y[[1]]
```

```
[1] 1 2
```

```
Y[[2]][2,]
```

```
[1] 2 20
```

```
Y[[3]][, , 2]
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	11	13	15	17	19
[2,]	12	14	16	18	20

```
Y[[3]][1, , 3]
```

```
[1] 21 23 25 27 29
```

Listen: Indizierung nach *Name*

Die Elemente einer Liste können benannt werden; die Namen können dann bei der Indizierung verwendet werden.

```
x <- c(1,2)
```

```
X <- matrix(c(1,2,3, 10,20,30), nrow = 3)
```

```
XX <- array(1:30, dim=c(2,5,3))
```

```
Y <- list(vek=x,mat=X,arr=XX)
```

```
Y[["vek"]] # oder analog:
```

```
Y$vek
```

```
Y[["mat"]] # oder analog:
```

```
Y$mat
```

```
names(Y) # Namen auslesen
```

```
names(Y) <- c("Aa", "Be", "Ce") # umbenennen
```

Listen: Anwendungsbeispiel

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  erg1 <- Befehl1  
  erg2 <- Befehl2  
  return(list(erg1,erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1 <- (x-mean(x))^ 2  
  V1 <-  
sum(b1) / (length(x)-1)  
  V2 <- var(x)  
  return(list(V1,V2))  
}  
myvarianz1(x=data)
```

Alternativ mit benannten
Listenobjekten:

```
myvarianz2 <- function(x)  
{  
  b1 <- (x-mean(x))^ 2  
  V1 <- sum(b1)/(length(x)-1)  
  V2 <- var(x)  
  return(list(myvarianz=V1,Rvarianz=V2))  
}  
myvarianz2(x=data)
```

Listen: Anwendungsbeispiel

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  erg1 <- Befehl1  
  erg2 <- Befehl2  
  return(list(erg1, erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1 <- (x-mean(x))^ 2  
  V1 <-  
sum(b1) / (length(x)-1)  
  V2 <- var(x)  
  return(list(V1, V2))  
}  
myvarianz1(x=data)
```

Alternativ mit benannten
Listenobjekten:

```
myvarianz2 <- function(x)  
{  
  b1 <- (x-mean(x))^ 2  
  V1 <- sum(b1)/(length(x)-1)  
  V2 <- var(x)  
  return(list(myvarianz=V1, Rvarianz=V2))  
}  
myvarianz2(x=data)
```


Listen: Anwendungsbeispiel

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  erg1 <- Befehl1  
  erg2 <- Befehl2  
  return(list(erg1, erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1 <- (x-mean(x))^2  
  V1 <-  
sum(b1) / (length(x)-1)  
  V2 <- var(x)  
  return(list(V1, V2))  
}  
myvarianz1(x=data)
```

Alternativ mit benannten
Listenobjekten:

```
myvarianz2 <- function(x)  
{  
  b1 <- (x-mean(x))^2  
  V1 <- sum(b1)/(length(x)-1)  
  V2 <- var(x)  
  return(list(myvarianz=V1, Rvarianz=V2))  
}  
myvarianz2(x=data)
```

Listen: Anwendungsbeispiel

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  erg1 <- Befehl1  
  erg2 <- Befehl2  
  return(list(erg1, erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1 <- (x-mean(x))^2  
  V1 <-  
sum(b1) / (length(x)-1)  
  V2 <- var(x)  
  return(list(V1, V2))  
}  
myvarianz1(x=data)
```

Alternativ mit benannten
Listenobjekten:

```
myvarianz2 <- function(x)  
{  
  b1 <- (x-mean(x))^2  
  V1 <- sum(b1)/(length(x)-1)  
  V2 <- var(x)  
  return(list(myvarianz=V1, Rvarianz=V2))  
}  
myvarianz2(x=data)
```

Listen: Anwendungsbeispiel

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  erg1 <- Befehl1  
  erg2 <- Befehl2  
  return(list(erg1, erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1 <- (x-mean(x))^2  
  V1 <-  
sum(b1) / (length(x)-1)  
  V2 <- var(x)  
  return(list(V1, V2))  
}  
myvarianz1(x=data)
```

Alternativ mit benannten
Listenobjekten:

```
myvarianz2 <- function(x)  
{  
  b1 <- (x-mean(x))^2  
  V1 <- sum(b1)/(length(x)-1)  
  V2 <- var(x)  
  return(list(myvarianz=V1, Rvarianz=V2))  
}  
myvarianz2(x=data)
```

Listen: Anwendungsbeispiel

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  erg1 <- Befehl1  
  erg2 <- Befehl2  
  return(list(erg1, erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1 <- (x-mean(x))^2  
  V1 <-  
sum(b1) / (length(x)-1)  
  V2 <- var(x)  
  return(list(V1, V2))  
}  
myvarianz1(x=data)
```

Alternativ mit benannten
Listenobjekten:

```
myvarianz2 <- function(x)  
{  
  b1 <- (x-mean(x))^2  
  V1 <- sum(b1)/(length(x)-1)  
  V2 <- var(x)  
  return(list(myvarianz=V1, Rvarianz=V2))  
}  
myvarianz2(x=data)
```

Übung 2.2

1) Schreiben Sie eine Funktion, die die dritte und vierte Wurzel aus den Werten eines Vektors (mit positiven Einträgen) berechnet und ausgibt.

2) Schreiben Sie eine Funktion, die die m -te und n -te (m und n beliebig aus N) Wurzel aus den Werten eines Vektors (mit positiven Einträgen) berechnet und ausgibt.

Die zentrale Datenstruktur: *data.frame*

- Listen nur mit Vektoren gleicher Länge als Elemente.
- `?data.frame`
- typischer Datentyp für Datensätze
- kann Spalten mit unterschiedlichen Datentypen haben (aber jede Spalte nur ein Datentyp).
- Indizierung wie bei Matrizen und Listen

	Geschl	Gew	Gr	Alt
1	M	70	168	20
2	M	65	172	19
3	M	80	186	24
4	W	66	172	19
5	W	59	170	22

data.frame definieren

```
stud <- data.frame(Geschl = c(rep("M", 3), rep("W", 2)),  
                  Gew = c(70, 65, 80, 66, 59),  
  
                  Alt = c(20, 19, 24, 19, 22))  
  
# oder schrittweise definieren:  
a <- c(rep("M", 3), rep("W", 2))  
b <- c(70, 65, 80, 66, 59)  
c <- c(168, 172, 186, 172, 170)  
d <- c(20, 19, 24, 19, 22)  
stud <- data.frame(a, b, c, d)  
colnames(stud) <- c("Geschl", "Gew", "Gr", "Alt")  
str(stud) # gibt die Struktur der Daten aus
```

data.frame: wichtige Befehle

<code>stud[, 2]</code>	2. Spalte auswählen
<code>stud[, "Gew"]</code>	2. Spalte auswählen
<code>stud\$Gew</code>	2. Spalte auswählen
<code>stud[, c(1, 3)]</code>	1. und 3. Spalte auswählen
<code>stud[-c(1, 4),]</code>	alle Spalten ohne 1. und 4. Zeilen
<code>names(stud)</code>	Spaltennamen ausgeben
<code>dim(stud)</code>	Dimensionen bestimmen
<code>mean(stud\$Gew)</code> <code>mean(stud[, 2])</code>	Mittelwert der 2. Spalte

data.frame: Teilmenge bilden

`subset` (Datensatzname, logischer Ausdruck); Siehe
`?subset`

```
stud.leicht <- subset(stud, stud$Gew <= 60)
stud.schwer <- subset(stud, Gew > 70)
stud.m <- subset(stud, Geschl=="M")
stud.m.gross <- subset(stud, Geschl=="M" & Gr > 180)
stud.in <- subset(stud, Alt %in% c(19,20))
```

oder direkt: Datensatzname[logischer Ausdruck,]

```
stud.leicht <- stud[stud$Gew <= 60 ,]
stud.schwer <- stud[stud[,2] > 70 ,]
stud.m <- stud[stud$Geschl=="M" ,]
stud.m.gross <- stud[stud[,1]=="M" & stud[,3] > 180 ,]
stud.in <- stud[stud$Alt %in% c(19,20) ,]
```

subset()

Description

Return subsets of vectors, matrices or data frames which meet conditions.

Usage

```
subset(x, subset, select, ...)
```

Arguments

`x` object to be subsetted.

`subset` logical expression indicating elements or rows to keep.

`select` expression, indicating columns to select from a data frame

subset()

Beispiel:

```
stud1 <- subset(stud, subset=stud$Gew <= 60)
stud2 <- subset(stud, subset=(Geschl=="M" & Gr > 180))
stud3 <- subset(stud, subset=(Alt %in% c(19,20)))
stud4 <- subset(stud, subset=(Geschl=="M"),
               select=c("Gew", "Alt"))
stud5 <- subset(stud, subset=(Geschl=="M"),
               select=-c(1,3))
identical(stud4, stud5)
```

Übung 2.3

Würfeln Sie 1000 mal und speichern Sie die Ergebnisse im Vektor x .
Speichern Sie die Ergebnisse kleiner oder gleich 5 im Vektor y und
alle *Sechser* in z und berechnen Sie wie oft ein *Sechser* geworfen
wurde?

Übung 2.4

Laden Sie mit dem Befehl `data(airquality)` den Datensatz *airquality* aus dem Paket *datasets* und lesen Sie die Hilfe zu diesem Datensatz.

- a Erstellen Sie einen neuen Datensatz *air1* mit *Wind* kleiner als 10.
- b Erstellen Sie den Datensatz *air2* mit *Temp* größer als 70 (F).
- c Der Datensatz *air3* soll so erstellt werden, dass er nur die Angaben zu den Variablen/Spalten *Month* und *Ozone*, bei denen *Wind* kleiner als 10 und *Temp* größer als 70 (F) sind, beinhaltet.

data.frame: Daten ordnen

`dataname[order(dataname[, spaltennummer]),]`; Siehe
`?order`

```
stud1 <- stud[order(stud[, 2]), ] # oder
stud1 <- stud[order(stud[, "Gew"]), ] # oder
stud1 <- stud[order(stud$Gew), ]

stud2 <- stud[order(stud$Gr), ]
stud3 <- stud[order(stud$Alt), ]
stud4 <- stud[order(stud$Gr, stud$Gew), ]
stud5 <- stud[order(stud$Gr, -stud$Gew), ]
```

Übung 2.5

Laden Sie den Datensatz *airquality* aus dem Paket *datasets*. Die Daten sind nach *Monat* und *Tag* sortiert.

- a Sortieren Sie die Daten nach Temperatur (Temp) aufsteigend.
- b Sortieren Sie die Daten nach Windstärke (Wind) zuerst absteigend und dann aufsteigend und überprüfen Sie ob die beiden Datensätze identisch sind.
- c Sortieren Sie die Daten nach Sonneneinstrahlung (Solar.R) absteigend. Benutzen Sie die verschiedenen Optionen des Arguments *na.last*.
- d Sortieren Sie die Daten nach Sonneneinstrahlung aufsteigend und nach Ozonkonzentration absteigend.

Lesen Sie in der Hilfe zu `order()` über das Argument `na.last`.

Eigenschaften verschiedener Datenstrukturen

<i>Objekt</i>	<i>Mögliche Typen</i>	<i>Mixen von Typen möglich</i>	<i>Spalten gleicher Länge</i>
<i>vector</i>	<i>numeric, character complex</i>	<i>nein</i>	<i>-</i>
<i>factor</i>	<i>(numeric), character</i>	<i>nein</i>	<i>-</i>
<i>matrix</i>	<i>numeric, character complex</i>	<i>nein</i>	<i>ja</i>
<i>array</i>	<i>numeric, character complex</i>	<i>nein</i>	<i>ja</i>
<i>list</i>	<i>numeric, character complex</i>	<i>ja</i>	<i>nein</i>
<i>data.frame</i>	<i>numeric, character complex</i>	<i>ja</i>	<i>ja</i>

Umwandlung verschiedener Datenstrukturen

Umwandlung verschiedener Datenstrukturen ist unter bestimmten Bedingungen möglich und sinnvoll:

Matrix \rightsquigarrow *data.frame* : `as.data.frame()`

data.frame \rightsquigarrow *Matrix* : `as.matrix()`

data.frame \rightsquigarrow *list* : `as.list()`

```
X <- matrix(c(1,2,3, 10,20,30), nrow = 3)
is.matrix(X); is.data.frame(X)
X1 <- as.data.frame(X)
X2 <- as.matrix(X1)
X3 <- as.list(X1)
X4 <- as.data.frame(X3)
```

Weitere logische Operationen zur Indizierung

```
any(), all(), which(), which.max(), which.min(), is.na(),  
!is.na(), is.nan()
```

.....

```
stud <- data.frame(Geschl = c(rep("M", 3), rep("W", 2)),  
                  Gew = c(70, 65, 80, 66, 59),  
                  Gr=c(168, 172, 186, 172, 170),  
                  Alt=c(20, 19, 24, 19, 22))
```

.....

```
any(stud$Gew > 100)
```

```
FALSE
```

```
all(stud$Alt <= 24)
```

```
TRUE
```

```
which(stud$Geschl == "W")
```

```
4 5
```

```
stud[which(stud[,3] <= 170),]
```

	Geschl	Gew (kg)	Gr (cm)	Alt
1	M	70	168	20
5	W	59	170	22

```
stud[which(stud[,1] != "M"),]
```

	Geschl	Gew (kg)	Gr (cm)	Alt
4	W	66	172	19
5	W	59	170	22

```
stud[which.max(stud[,3]),]
```

	Geschl	Gew (kg)	Gr (cm)	Alt
3	M	80	186	24

```
x<-c(1, 4, NA)
```

```
x[!is.na(x)]
```

```
1 4
```

Übung 2.6

Laden Sie den Datensatz *airquality* aus dem Paket *datasets*.

- a Bereinigen Sie den Datensatz um fehlende Werte und legen Sie ihn als Variable *air3* im Speicher ab. Hinweis: `na.omit()` oder `complete.cases()`.
- b Bereinigen Sie den Datensatz um fehlende Werte der 1. Spalte und speichern Sie ihn als Variable *air4* (Hinweis: `!is.na()`).
- c Bereinigen Sie den Datensatz um fehlende Werte der 1. und 2. Spalte und speichern Sie ihn als Variable *air5*.
- d Wann war der Wind (bzw. die Temperatur, Ozonkonzentration, Sonneneinstrahlung) am stärksten?

Data Frame: Hinzufügen neuer Variablen

Datensatz stud (von vorhin) um neue Variable erweitern:

```
stud$BMI <- (stud$Gew) / (stud$Gr/100) ^2 ;  
stud
```

	Geschl	Gew (kg)	Gr (cm)	Alt	BMI
1	M	70	168	20	24.80159
2	M	65	172	19	21.97134
3	M	80	186	24	23.12406
4	W	66	172	19	22.30936
5	W	59	170	22	20.41522

Alternativ: neue Variable *BMI* definieren,

```
BMI<-(stud$ Gew) / ((stud$ Gr/100) ^ 2))
```

und mittels *cbind()* dem Datensatz als weitere Spalte hinzuzufügen:

```
stud<-cbind(stud,BMI) ; stud
```

Übung 2.7

Fügen Sie dem Datensatz *stud* eine neue Variable hinzu, die das $(\text{Gewicht/Groesse}) \cdot \text{Alter}$ angibt und sortieren Sie dann den Datensatz nach der neuen Variable.

Übung 2.7

Fügen Sie dem Datensatz *stud* eine neue Variable hinzu, die das $(\text{Gewicht}/\text{Groesse}) \cdot \text{Alter}$ angibt und sortieren Sie dann den Datensatz nach der neuen Variable.

```
stud$var1 <- (stud$Gew/stud$Gr) * stud$Alt
stud
stud[order(stud$var1), ]
```

Erinnerung: Funktionen definieren

Zusätzlich zu bereits definierten Funktionen in  (z.B. `sum`, `min`) kann man eigene Funktionen definieren. Hier ein Beispiel:

```
meine.fun <-function(x1 ,x2 ) {  
  y <- (x1 + x2 )/2  
  y  
}
```

Nachdem man die Funktion `meine.fun` definiert hat, kann man sie aufrufen:

```
meine.fun(100,200)
```

```
150
```

oder

```
meine.fun(x1=100, x2=200)
```

oder

```
meine.fun(x2=200, x1=100)
```

Nochmal zu Funktionen: Rückgabewert

```
Funktionsname <- function(arg1, arg2) {  
  statements  
}
```

Das letzte erzeugte Objekt wird zurückgegeben.

```
Funktionsname <- function(arg1, arg2) {  
  statements  
  return(object)  
}
```

Das Objekt *object* wird zurückgegeben.

Beispiel: Varianz berechnen

Setzen Sie die Formel für die Varianz

$$\text{var}(x) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

als Funktion in  um!

```
myvarianz <- function(x) {  
  a1<-mean(x)  
  a2<-length(x)  
  a3<-(x-a1)^2  
  a4<-sum(a3)  
  a5<-a4/(a2-1)  
  return(a5) # oder a5  
}  
data <- seq(0,1000,0.1)  
myvarianz(x=data)
```

Nochmal zu Funktionen: mehrere Rückgabewerte

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  Befehl1  
  Befehl2  
  return(list(erg1, erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1)/(length(x)-1)  
  V2<-var(x)  
  return(list(V1,V2))  
}  
myvarianz1(x=data)
```

oder den Objekten in der Liste
Namen geben:

```
myvarianz2<-function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1)/(length(x)-1)  
  V2<-var(x)  
  return(list(myvarianz=V1,  
              Rvarianz=V2))  
}  
myvarianz2(x=data)
```

Nochmal zu Funktionen: mehrere Rückgabewerte

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  Befehl1  
  Befehl2  
  return(list(erg1, erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1)/(length(x)-1)  
  V2<-var(x)  
  return(list(V1,V2))  
}  
myvarianz1(x=data)
```

oder den Objekten in der Liste
Namen geben:

```
myvarianz2<-function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1)/(length(x)-1)  
  V2<-var(x)  
  return(list(myvarianz=V1,  
              Rvarianz=V2))  
}  
myvarianz2(x=data)
```

Nochmal zu Funktionen: mehrere Rückgabewerte

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  Befehl1  
  Befehl2  
  return(list(erg1, erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1)/(length(x)-1)  
  V2<-var(x)  
  return(list(V1,V2))  
}  
myvarianz1(x=data)
```

oder den Objekten in der Liste
Namen geben:

```
myvarianz2<-function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1)/(length(x)-1)  
  V2<-var(x)  
  return(list(myvarianz=V1,  
              Rvarianz=V2))  
}  
myvarianz2(x=data)
```

Nochmal zu Funktionen: mehrere Rückgabewerte

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  Befehl1  
  Befehl2  
  return(list(erg1, erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1) / (length(x)-1)  
  V2<-var(x)  
  return(list(V1,V2))  
}  
myvarianz1(x=data)
```

oder den Objekten in der Liste
Namen geben:

```
myvarianz2<-function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1) / (length(x)-1)  
  V2<-var(x)  
  return(list(myvarianz=V1,  
              Rvarianz=V2))  
}  
myvarianz2(x=data)
```

Nochmal zu Funktionen: mehrere Rückgabewerte

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  Befehl1  
  Befehl2  
  return(list(erg1, erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1)/(length(x)-1)  
  V2<-var(x)  
  return(list(V1,V2))  
}  
myvarianz1(x=data)
```

oder den Objekten in der Liste
Namen geben:

```
myvarianz2<-function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1)/(length(x)-1)  
  V2<-var(x)  
  return(list(myvarianz=V1,  
              Rvarianz=V2))  
}  
myvarianz2(x=data)
```

Nochmal zu Funktionen: mehrere Rückgabewerte

Funktion mit mehreren Ausgaben:

```
function(arg1, arg2) {  
  Befehl1  
  Befehl2  
  return(list(erg1, erg2))  
}
```

Beispiel:

```
myvarianz1 <- function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1)/(length(x)-1)  
  V2<-var(x)  
  return(list(V1,V2))  
}  
myvarianz1(x=data)
```

**oder den Objekten in der Liste
Namen geben:**

```
myvarianz2<-function(x)  
{  
  b1<-(x-mean(x))^ 2  
  V1<-sum(b1)/(length(x)-1)  
  V2<-var(x)  
  return(list(myvarianz=V1,  
              Rvarianz=V2))  
}  
myvarianz2(x=data)
```

Variadische Funktionen schreiben

Mittels dreier Auslassungspunkte "..." kann man die genaue Anzahl an erwarteten Argumenten offenlassen.

Beispiel:

```
foo <- function(...)
print(as.list(match.call())) # listet alle
Argumente auf
foo(a=1, b=2, c=3, d=4)
```

Praktische Anwendung: Argumente an innere Funktion weitergeben

```
plot.blau <- function(x, y, ...) {
plot(x, y, col="blue", ...)
}
x1<-1:10
y1<-x1^ 2
plot.blau(x=x1, y=y1, xlim=c(0, 15),
xlab="X-values")
```


Nochmal zu Funktionen: Übung

Übung 2.8

- 1) Schreiben Sie eine Funktion, welche die dritte und vierte Wurzel aus den Werten eines Vektors (mit positiven Einträgen) berechnet und zurückgibt.
- 2) Schreiben Sie eine Funktion, welche die m -te und n -te (m und n beliebig aus N) Wurzel aus den Werten eines Vektors (mit positiven Einträgen) berechnet und zurückgibt.

Lösung

```
wurzelB <- function(x) {  
  wurzel3 <- x^(1/3)  
  wurzel4 <- x^(1/4)  
  return(list(wurzel3=wurzel3, wurzel4=wurzel4))  
}
```

```
wurzel.mn <- function(x, m=3, n=4) {  
  wurzelm <- x^(1/m)  
  wurzeln <- x^(1/n)  
  return(list(wurzelm=wurzelm, wurzeln=wurzeln))  
}
```

Nochmal zu Funktionen: Anweisungsblock { }

```
function() { Befehle }
```

Man **kann** Funktionen ohne Anwendung eines Anweisungsblocks definieren, wenn sie aus einem einzigem Befehl bestehen.

```
function() Befehl # Beispiel:
```

```
my.mean <- function(x) sum(x) / length(x)  
# oder  
my.mean <- function(x) {  
  sum(x) / length(x)  
}
```

Dies gilt für alle Kontrollstrukturen, die wir später lernen.

Kontrollstrukturen

sind in  z.B. bedingte Anweisungen und Schleifen.

Einführendes Beispiel:

```
x <- 2.5  
if (x <= 0) { 0 } else { x }
```

2.5

Das ist also die bedingte Anweisung:

WENN x kleiner oder gleich 0

DANN 0 zurückgeben

SONST x zurückgeben

Bedingte Anweisungen: *if ... else ...* I

1. `if (Bedingung) { Befehlsfolge 1 } * else { Befehlsfolge 2}.`

Die Bedingung kann ein komplexerer logischer Ausdruck sein.

```
x <- 999
if (x < 0|x == 999) { x <- NA} else { x <- x }
x
NA
```

*****: kein Zeilenumbruch hier

oder:

```
x <- 999
if ( x < 0|x == 999) {
x <- NA} else {
x <- x }
x
```

oder:

```
x <- 999
if ( x < 0|x == 999) {x <- NA} else {
x <- x }
x
```

Bedingte Anweisungen: *if ... else ...* I

1. `if (Bedingung) { Befehlsfolge 1 } * else { Befehlsfolge 2}`.
Die Bedingung kann ein komplexerer logischer Ausdruck sein.

```
x <- 999
if (x < 0|x == 999) { x <- NA} else { x <- x }
x
NA
```

*: kein Zeilenumbruch hier

oder:

```
x <- 999
if ( x < 0|x == 999) {
x <- NA} else {
x <- x }
x
```

oder:

```
x <- 999
if ( x < 0|x == 999) {x <- NA} else {
x <- x }
x
```

Bedingte Anweisungen: *if ... else ...* I

1. `if (Bedingung) { Befehlsfolge 1 } * else { Befehlsfolge 2}`.
Die Bedingung kann ein komplexerer logischer Ausdruck sein.

```
x <- 999
if ( x < 0 | x == 999) { x <- NA } else { x <- x }
x
NA
```

* : kein Zeilenumbruch hier

oder:

```
x <- 999
if ( x < 0 | x == 999) {
x <- NA } else {
x <- x }
x
```

oder:

```
x <- 999
if ( x < 0 | x == 999) {x <- NA} else {
x <- x }
x
```

Bedingte Anweisungen: *if ... else ...* I

1. `if (Bedingung) { Befehlsfolge 1 } * else { Befehlsfolge 2}`.
Die Bedingung kann ein komplexerer logischer Ausdruck sein.

```
x <- 999
if ( x < 0 | x == 999) { x <- NA } else { x <- x }
x
NA
```

* : kein Zeilenumbruch hier

oder:

```
x <- 999
if ( x < 0 | x == 999) {
x <- NA } else {
x <- x }
x
```

oder:

```
x <- 999
if ( x < 0 | x == 999) {x <- NA } else {
x <- x }
x
```


Bedingte Anweisungen: *if ... else ...* II

statt *if (Bedingung) { Befehlsfolge 1 } else { Befehlsfolge 2}*:

if (Bedingung) Befehl 1 else Befehl 2

Man **kann** bedingte Anweisungen ohne Anwendung eines Anweisungsblocks definieren, wenn sie aus einem einzigem Befehl besteht.

Beispiel:

```
x <- 999  
if ( x < 0 | x == 999) x <- NA else x <- x  
x
```

NA

Bedingte Anweisungen: *if ... else ...* III

if else kann nur auf Skalaren (d.h. Vektoren der Länge 1) angewandt werden:

Beispiel 2.1

Schreiben Sie eine **Funktion**, welche den Wert 999 durch *NA* ersetzt.
(x hat nur einen Eintrag!)

.....

Bedingte Anweisungen: *if ... else ...* III

if else kann nur auf Skalaren (d.h. Vektoren der Länge 1) angewandt werden:

Beispiel 2.1

Schreiben Sie eine **Funktion**, welche den Wert 999 durch *NA* ersetzt.
(x hat nur einen Eintrag!)

.....

```
fun1 <- function(x) {  
  if (x==999) {  
    NA } else {  
    x  
  }  
}
```

Bedingte Anweisungen: *if ... else ...* III

if else kann nur auf Skalaren (d.h. Vektoren der Länge 1) angewandt werden:

Beispiel 2.1

Schreiben Sie eine **Funktion**, welche den Wert 999 durch *NA* ersetzt. (x hat nur einen Eintrag!)

.....

```
fun1 <- function(x) {  
  if (x==999) {  
    NA } else {  
    x  
  }  
}
```

oder:

```
fun1 <- function(x) {  
  if (x==999) NA else x  
}
```

Bedingte Anweisungen: *ifelse()*

2. Auch die Funktion **ifelse()** aus Übungsblatt 3 ist eine bedingte Anweisung:

ifelse(Bedingung, Ausdruck1, Ausdruck2)

Dient der **vektorwertigen** Auswertung von Bedingungen, d.h. *ifelse* kann auf Vektoren mit mehreren Einträgen angewandt werden.

Beispiel:

```
x <- c(1, -pi, log(0.25), 10, NA)
```

```
y <- ifelse(x < 0, NA, x); y
```

```
1, NA, NA, 10, NA
```

if ... else ... if ... else

Nach **else** kann wieder eine neue bedingte Anweisung folgen:

```
if ( Bedingung1) { Befehlsfolge 1 } else if ( Bedingung2) { Befehlsfolge 2 } else { Befehlsfolge 3 }
```

```
x <- 12
if (x<=0) {
  y <- 0
} else if (x>0 & x<=1) {
  y <- x
} else {
  y <- 1
}
y
```

1

any und *all*

```
x <- c(1,-1,101,4)
if (any (x < 0)) {
  print("negative_Werte_in_x")
} else {
  print("keine_neg._Werte")
}

"negative Werte in x"
```

```
mein.fun <- function(x) {
  if (any (x < 0 )) {
    print("negative_Werte_in_x")
  } else {
    print("keine_neg._Werte")
  }
}

mein.fun(x=c(2,3,99))

"keine neg. Werte"
```

Übung 2.9

Schreiben Sie eine neue Funktion mit Hilfe von *all()*, die das gleiche tut!

Übung 2.10

Schreiben Sie eine neue Funktion, die angibt ob es in einer Spalte eines Datensatzes *NA-values* gibt!

Schleifen

Um Iterationen durchzuführen gibt es in  **R**:

- a. *for*-Schleifen
- b. *while*-Schleifen
- c. *repeat*-Schleifen

Schleifen bestehen aus

- **Schleifenkontrolle**, in welcher entschieden wird ob eine Iteration durchgeführt wird
- **Anweisungsfolgen**, welche in jeder Iteration durchgeführt werden

z.B.:

for(Schleifenkontrolle) { Anweisungsfolgen }

Schleifen: *for*-Schleifen

Typischer Aufbau:

for (Zählvariable in Menge) { Anweisungsfolge }

Beispiel 1:

```
x <- c(4,10,11,12)
for (i in 1:4) {
  print(x[i]^2)
}
```

16

100

121

144

Schleifen: *for*-Schleifen

Typischer Aufbau:

for (Zählvariable in Menge) { Anweisungsfolge }

Beispiel 1:

```
x <- c(4,10,11,12)
for (i in 1:4){
  print(x[i]^2)
}
```

16
100
121
144

Beispiel 2:

```
x <- c(4,10,11,12)
y <- NULL
for (i in 1:length(x)) {
  y[i] <- x[i]^2
}
y
```

16 100 121 144

Schleifen: *for*-Schleifen

Typischer Aufbau:

for (Zählvariable in Menge) { Anweisungsfolge }

Beispiel 1:

```
x <- c(4,10,11,12)
for (i in 1:4){
  print(x[i]^2)
}
```

16
100
121
144

Beispiel 2:

```
x <- c(4,10,11,12)
y <- NULL
for (i in 1:length(x)) {
  y[i] <- x[i]^2
}
```

y

16 100 121 144

Eigentlich braucht man hier kein Schleife:

```
x <- c(4,10,11,12)
z <- x^2
```

Schleifen: *for*-Schleifen – Beispiel

Beispiel 2.2

Die Merkmale *Ozone* und *Solar.R* im Datensatz *airquality* haben Einträge mit *NA*.

- a. Fügen Sie dem Datensatz eine neue Variable *Ozone1* hinzu, welche anstelle der *NA*-Einträge der *Ozone*-Spalte den Wert -9999 enthält.
- b. Definieren Sie eine neue Spalte, die in jeder Zeile „kontrolliert“, ob irgendein Merkmal den fehlenden Wert *NA* hat.

Lösen Sie diese Aufgabe mit Hilfe von for-Schleifen

Schleifen: *for*-Schleifen – Beispiel

Lösung zu a:

```
Ozone1 <- NULL # definiere eine leere Menge (Vektor)
# oder mit Ozone1 <- c()
n <- nrow(airquality) # Anzahl der Zeilen
for (i in 1:n){
  if(is.na(airquality[i,1])==TRUE){
    Ozone1[i] <- 9999
  } else {
    Ozone1[i] <- airquality[i,1]
  }
}
Air1 <- airquality
Air1$Ozone1 <- Ozone1
```

Schleifen: *for*-Schleifen – Beispiel

Lösung zu b:

```
control <- NULL
n <- nrow(airquality)
for (i in 1:n){
  if(any(is.na(airquality[i,]))==TRUE){
    control[i] <- 9999
  } else {
    control[i] <- 0
  }
}
Air2 <- airquality
Air2$control <- control
```

Übung 2.11

Lösen Sie die Aufgabe im Beispiel 2.2 mit Hilfe von `ifelse()`.

Schleifen: *while*

Bei for-Schleifen steht im Voraus fest, wie oft die Schleife wiederholt wird. (selbst falls die Zählvariable innerhalb der Schleife geändert wird...)
while-Schleifen funktionieren auch bei unbestimmter Iterationsanzahl:

while (Bedingung) { Anweisungsfolge }

while-Beispiel:

```
i <- 0
summe <- 0
while (summe <= 50000) {
  summe <- summe+i
  i <- i+1 }
summe
```

50086

i

317

Schleifen: *while*

Bei for-Schleifen steht im Voraus fest, wie oft die Schleife wiederholt wird. (selbst falls die Zählvariable innerhalb der Schleife geändert wird...)
while-Schleifen funktionieren auch bei unbestimmter Iterationsanzahl:

while (Bedingung) { Anweisungsfolge }

while-Beispiel:

```
i <- 0
summe <- 0
while (summe <= 50000) {
  summe <- summe+i
  i <- i+1 }
summe
```

50086

i

317

Gleiches Beispiel mit *for*:

```
summe <- 0
for (i in 1:316) {
  summe <- summe+i
}
summe
```

50086

Schleifen: *while* – Beispiel

Übung 2.12

Wie ersetzt man im Datensatz `airquality` mithilfe einer *while*-Schleife alle *NA*-Einträge (in allen Spalten) mit 9999?

Schleifen: *repeat*

Quasi *umgedrehte* while-Schleifen; *zuerst* werden die *Anweisungen* abgearbeitet, *dann* wird die *Bedingung* geprüft:

```
repeat { Anweisungsfolge  
Bedingung { break } }
```

```
i <- 0  
Summe <- 0  
repeat {  
  Summe <- Summe+i  
  i <- i+1  
  if (Summe > 50000) {  
    break }  
}  
Summe
```

50086

Im Gegensatz zu *while* werden bei der **repeat**-Schleife die Anweisungen **mindestens einmal** ausgeführt!

split()

Zertrennen eines Datensatzes in mehrere Datensätze – sodass für jede mögliche Ausprägung einer bestimmten Variablen ein eigener Datensatz entsteht:

split(x, fac, ...)

Beispiel 2.3

Laden Sie den Datensatz [Orange](#) aus dem Paket [datasets](#) und lesen Sie die Informationen zu diesem Datensatz aus dem Netz. Der Datensatz enthält die Umfänge von 5 Orangenbäumen in 7 verschiedenen Altern. Bilden Sie 5 neue Datensätze aus diesem Datensatz für den jeweiligen Baum. Bilden Sie 7 Datensätze aus [Orange](#), die jeweils die Einträge der Bäume mit gleichem Alter enthalten.

Lösung

```
# Zuerst ohne split-Funktion, sondern mit Hilfe von subset():  
head(Orange)  
data1 <- subset(Orange, Orange$Tree==1)  
data2 <- subset(Orange, Orange$Tree==2)  
data3 <- subset(Orange, Orange$Tree==3)  
data4 <- subset(Orange, Orange$Tree==4)  
data5 <- subset(Orange, Orange$Tree==5)
```

Lösung

```
# Zuerst ohne split-Funktion, sondern mit Hilfe von subset():  
head(Orange)  
data1 <- subset(Orange, Orange$Tree==1)  
data2 <- subset(Orange, Orange$Tree==2)  
data3 <- subset(Orange, Orange$Tree==3)  
data4 <- subset(Orange, Orange$Tree==4)  
data5 <- subset(Orange, Orange$Tree==5)  
  
# oder mit split():  
datasets <- split(Orange, Orange$Tree)  
datasets # Datensätze sind in Liste kombiniert!
```

Lösung

```
# Zuerst ohne split-Funktion, sondern mit Hilfe von subset():  
head(Orange)  
data1 <- subset(Orange, Orange$Tree==1)  
data2 <- subset(Orange, Orange$Tree==2)  
data3 <- subset(Orange, Orange$Tree==3)  
data4 <- subset(Orange, Orange$Tree==4)  
data5 <- subset(Orange, Orange$Tree==5)
```

```
# oder mit split():  
datasets <- split(Orange, Orange$Tree)  
datasets # Datensätze sind in Liste kombiniert!
```

.....

```
data1 <- split(Orange, Orange$age)  
data1
```

Übung 2.13

Laden Sie den Datensatz `chickwts` aus dem Paket `datasets` und lesen Sie die Informationen zu diesem Datensatz. Der Datensatz enthält das Gewicht von 71 Hühnern welche jeweils einen von 7 verschiedenen Futtertypen erhalten.

1. Zerlegen Sie den Datensatz in je einen Datensatz pro Futtertyp.
2. Ermitteln Sie den Mittelwert des Gewichts für die jeweiligen Futtertypen.
3. Welches Futter bewirkt bei den Hühnern das maximale Gewicht? Welches bewirkt das minimale Gewicht?

apply()

Wendet eine Funktion (FUN) dimensionsweise (z.B. pro Zeile oder Spalte) auf ein Array X an!

```
apply(X, MARGIN, FUN)
```

X: ein Array (oder eine Matrix)

MARGIN: 1 (zeilenweise), 2 (spaltenweise), ... (scheibchenweise)

Beispiele:

```
dat <- matrix(1:6, ncol=2, byrow=T)
```

```
apply(dat, 1, mean)
```

```
apply(dat, 2, mean)
```

```
apply(airquality, 2, function(x) any(is.na(x)))
```

lapply()

Wendet eine Funktion (FUN) komponentenweise auf eine Liste, einen Datensatz oder einen Vektor X an!

`lapply(X, FUN)`

Beispiele:

```
lapply(airquality, function(x) any(is.na(x)))
```

```
# oder mit eigener Funktion:
```

```
fun1 <- function(x) any(is.na(x))
```

```
lapply(airquality, fun1)
```

```
lapply(airquality, mean)
```

```
lapply(airquality, mean, na.rm=T)
```

sapply()

Wendet eine Funktion (FUN) komponentenweise auf eine Liste, einen Datensatz oder einen Vektor X an! – Wie lapply(), nur der Output ist anders (Ausgabe als Vektor oder Matrix).

`sapply(X, FUN)`

Beispiele:

```
sapply(airquality, function(x) any(is.na(x)))
```

```
# oder:
```

```
fun1 <- function(x) any(is.na(x))
```

```
sapply(airquality, fun1)
```

```
sapply(airquality, mean)
```

```
sapply(airquality, mean, na.rm=TRUE)
```

tapply()

Wendet eine Funktion (FUN) auf X gruppiert nach INDEX an!

```
tapply(X, INDEX, FUN)
```

X: ein Vector

INDEX: Liste von einem oder mehreren Faktoren, alle von der gleichen Länge wie X

Beispiel:

```
tapply(airquality[,1], airquality[,5], mean, na.rm=TRUE)
```

aggregate()

Teilt Daten aus einem Data Frame in Teilmengen, berechnet eine Aggregationsfunktion dafür und gibt das Ergebnis übersichtlich zurück.

```
aggregate(X, by, FUN)
```


X: ein Data Frame

by: eine Liste von Gruppenelementen, alle jeweils so lang wie die Variablen im Data Frame x. Elemente werden automatisch in Faktoren umgewandelt.

Beispiel: Vergleichen Sie die unterschiedlichen Ergebnisse von `tapply()` und `aggregate()`

```
aggregate(airquality$Temp, list(airquality$Month), mean)  
tapply(airquality$Temp, list(airquality$Month), mean)
```

Eingabe/Einlesen und Ausgabe von Daten

1. Eingabe oder Einlesen von Daten
 - a. Direkte Eingabe
 - b. Daten Einlesen (importieren)
 - b.1 Text-Dateien
 - b.2 Binär-Dateien
 - b.3 -Dateien (*.RData*-Daten)
2. Ausgabe von Daten (exportieren)

Direkte Eingabe von Daten

► Wie bisher, z.B.:


```
# direkte Eingabe eines Vektors
x <- c(2, 5, 4)
# direkte Eingabe eines data.frame
stud <- data.frame(Geschl = c(rep("M", 3), rep("W", 2)),
                   Gew = c(70, 65, 80, 66, 59),
                   Gr=c(168, 172, 186, 172, 170),
                   Alt=c(20, 19, 24, 19, 22))
```

► Oder im eigenen Editor-Fenster

```
my.data <- data.frame()
fix(my.data) # oder
my.data <- edit(data.frame())
```

- 1) Auf Variablennamen (Tabellenkopf) klicken und Namen/Typ festlegen/ändern.
- 2) Am Ende den *Dateneditor* schliessen.

Übung 2.14

Erzeugen Sie mit Hilfe des -Editors einen Datensatz, der das Geschlecht, die Groesse und das Gewicht von 5 Personen enthält. Nachdem Sie den Datensatz abgespeichert haben, versuchen Sie einige Einträge zu ändern.

Daten Importieren

- Meist werden Daten vom Arbeitsverzeichnis oder aus dem Netz importiert. Je nach Format des Datensatzes verwendet man Befehle wie: *read.csv()*, *read.table()*.

```
data1 <- read.table(file="D:/Einfuerung_R/gewicht.txt")
```

- Man kann den Pfad *absolut* angeben, z.B.
file="D:/Einfuerung_R/Daten/gewicht.txt"
oder *relativ*, z.B. wenn die Datei im Arbeitsverzeichnis liegt:
file="gewicht.txt".
- Wechseln des Arbeitsverzeichnisses siehe S.69.
- Wahl einer Datei per Dialogbox:

```
data1 <- read.table(file=file.choose())
```

- beachte Groß- und Kleinbuchstaben bei Pfadangaben (unter Windows egal...).

Tabellen aus Textdateien einlesen

- Tabellen im Textformat (meist *.txt* oder *.csv*) enthalten Zeilen und Spalten in Form von Buchstaben, Zahlen, Sonderzeichen und Steuerzeichen. – Der einfachste Weg, relativ kleine Datenmengen von einem System in ein anderes zu transportieren (für Latin-1 oder UTF-8 kodierten Text, sonst re-encoding mit dem Argument **fileEncoding**).
- Lesen einer Tabelle – Syntax:

```
read.table(file, header =FALSE, sep = " ", quote=" \"' ",  
dec = ". ", row.names, col.names, na.strings = "NA", ... )
```

Beispiele:

```
data2 <- read.table(file="D:/Einfuerung_R/gewicht.txt")  
data3 <- read.table(file="http://statland.org/R/R/basketball.txt",  
                    header=T)
```

read.table(): wichtige Argumente

file: Pfad und Dateiname; Pfade entweder mit Slash (/) oder doppeltem Backslash (\\).

header: Erste Zeile als Spaltennamen verwenden (default: FALSE)

sep: Trennzeichen zwischen den Spalten (default: " "; ein/mehrere Leerzeichen oder Tabulatoren).

quote: Angabe von Zeichenketten (default: "\"'"; es werden Anführungszeichen erkannt).

dec: Dezimalzeichen (default: ". ")

na.strings: Bezeichnung für fehlende Werte (default: "NA")

row/col.names: legt Namen der Zeilen bzw. Spalten fest.

colClasses: gibt an von welcher Klasse (Datentyp) die Spalten sind – v.a. bei großen Datensätzen, um zu verhindern, dass für jeden Tabelleneintrag die Konsistenz der Datentypen geprüft wird.

nrow: Anzahl maximal einzulesender Zeilen (default: -1; alle Zeilen einlesen).

skip: Anzahl der Zeilen, die am Beginn der Textdatei ignoriert werden. (default: 0; keine überspringen)

weitere Argumente: siehe *?read.table*

read.csv(), read.delim(), ...

Die Befehle *read.csv()*, *read.csv2()*, *read.delim()* und *read.delim2()* eignen sich ebenfalls für das Einlesen von Text-Dateien – jeweils mit gleicher Syntax (siehe ?read.csv, ...), aber anderen Voreinstellungen:

	sep=	dec=
read.table	" "	dec = " . "
read.csv	" , "	dec = " . "
read.csv2	" ; "	dec = " , "
read.delim	" \t "	dec = " . "
read.delim2	" \t "	dec = " , "

Binär-Daten Einlesen (Importieren)

Mit dem *foreign*-Paket ist es möglich Dateitypen anderer Statistik programmen einzulesen (*Minitab*, *SAS*, *SPSS*, *Stata*)

read.mtp(): Lesen von MiniTab Portable-Dateien

read.xport(): Lesen von SAS Transport-Dateien


read.S(): Lesen von S-Dateien

read.spss(): Lesen von SPSS-Dateien

read.dta(): Lesen von Stata-Dateien (bis Version 12)

Excel-Dateien lassen sich meist mit dem Package *readxl* – direkt aus der RStudio-Oberfläche heraus importieren.

R-Dateien (.RData) Einlesen

.RData und *.rda*-Dateien sind übliche Dateiendungen für das R-eigene Dateiformat. Mit dem Befehl *load()* kann eine -Datei geladen werden.

Lesen Sie im Abschnitt *Datenexport* wie man *R-Dateien* erzeugen kann.

```
mypath1 <- "C:/meinverzeichnis/"  
myfile1 <- paste(mypath1, "titanic.RData", sep="")  
load(file=myfile1) <
```

Mit *print(load(file=myfile1))* sieht man welche Objekte in "titanic.RData" enthalten sind.

Literatur: <http://cran.r-project.org/doc/manuals/R-data.pdf>

Datenexport

- 1 `save()`: als R-Datei (.RData bzw. .rda)
- 2 `write.table()`, `write.csv()`: als Text- oder csv-Datei
- 3 `write.foreign()`: als SPSS- und SAS- Datei
- 4 `write.xlsx`: als Excel-Datei (Package xlsx)

Beispiel:

```
x <- c(1, 2)
X <- matrix(c(1, 2, 3, 10, 20, 30), nrow = 3)
XX <- array(1:30, dim=c(2, 5, 3))
# 1. Die Daten als R-Datei exportieren:
save(x, X, XX, file="mysave.RData")
# 2. X als csv exportieren
write.table(X, file="X.txt")
# 3. X in SAS exportieren
library(foreign)
write.foreign(df=data.frame(X), datafile="X.csv",
              codefile="X.sas", package="SAS")
```

`write.foreign` erzeugt eine CSV-Datei und ein SAS- bzw. SPSS-Skript um die CSV-Datei einzulesen.

Datenim- und export

	Importieren	Exportierten	Format
ASCII	read.table()	write.table()	.txt, .csv
SAS	read.xport() (foreign)	write.xport() (SASxport)	.xpt, .dat
SPSS	read.spss()	write.foreign()	.sav, .dat
R-Datei	load()	save()	.RData, .rda
Excel-Datei	read.xlsx()	write.xlsx()	.xlsx

Weitere Im- und Exportmöglichkeiten sind über die Packages **readr** (Textdateien), **readxl** (Excel) und **haven** (SAS, SPSS, STATE) verfügbar und direkt über die RStudio-Oberfläche abrufbar.

Gliederung I

Organisatorisches

- | | |
|--|-----|
| 1. Einstieg und Grundlegendes zu  | 42 |
| 2. Datentypen und Datenimport | 92 |
| 3. Deskriptive Statistik mit <i>R</i> | 193 |
| 4. Verteilungen & Zufallszahlen in  | 349 |
| 5. Schliessende Statistik: Testen und Schätzen | 441 |

3. Deskriptive Statistik mit R

Mit Hilfe von

- I Tabellen
 - II Grafiken
 - III Charakteristische Maßzahlen
-
- a. Univariate Verfahren
 - b. Bi-/Multivariate Verfahren
-
- ▶ Nominalskalierte Merkmale (qualitativ)
 - ▶ Ordinalskalierte Merkmale (qualitativ)
 - ▶ Kardinalskalierte Merkmale (quantitativ)

3. Deskriptive Statistik für:

1. ein qualitatives Merkmal (univariate)
2. zwei qualitative Merkmale (bivariate)
3. ein quantitatives Merkmal (univariate)
4. ein qualitatives und ein quantitatives Merkmal (bivariate)
5. zwei quantitative Merkmale (bivariate)

Häufigkeitstabellen (1 qualitatives Merkmal)

Absolute Häufigkeiten:

```
table(..., exclude = if (useNA == "no") c(NA, NaN),  
useNA = "no", ...)
```

... : ein oder mehrere Objekte, die als Faktor interpretiert werden können (auch Zeichenketten) oder eine Liste bzw. ein Data.frame solcher Objekte.

Relative Häufigkeiten:

```
prop.table(x, margin = NULL)
```

x : eine Tabelle

Kumulative Häufigkeiten:

```
cumsum(x)
```

x : ein numerisches oder komplexes Objekt, bzw. eines das entsprechend umgewandelt werden kann.

Häufigkeitstabellen (absolute & relative)

table() , prop.table()

```
data1 <- rep(c("M", "W"), c(6, 4))
```

```
# absolute Häufigkeiten
```

```
tab1 <- table(data1)
```

```
tab1
```

```
M  W
```

```
6  4
```

```
# relative Häufigkeiten
```

```
prop.table(tab1)
```

```
M  W
```

```
0.6 0.4
```

Kumulative Häufigkeiten

Bezeichne a_1, a_2, \dots, a_k die der Größe nach geordneten, verschiedenen Merkmalsausprägungen des **ordinalskalierten** Merkmals X mit den entsprechenden absoluten Häufigkeiten: n_1, n_2, \dots, n_k .

Dann kann die kumulierte absolute (bzw. relative) Häufigkeit N_i (bzw. H_i) der Ausprägung a_i werden berechnet durch:

$$N_i = n_1 + n_2 + \dots + n_i = \sum_{j=1}^i n_j$$

$$H_i = h_1 + h_2 + \dots + h_i = \sum_{j=1}^i h_j$$

Kumulative Häufigkeiten: Beispiel

```
data2 <- rep(c("leicht", "normal", "schwer"), c(16, 40, 4))  
tab2 <- table(data2)
```

```
# kumulative absolute Häufigkeiten
```

```
cumsum(tab2)
```

```
leicht normal schwer
```

```
16      56      60
```

```
# kumulative relative Häufigkeiten
```

```
cumsum(prop.table(tab2))
```

```
leicht normal schwer
```

```
0.267  0.933  1
```

Kreis- Balkendiagramme (für 1 qualitatives Merkmal)

pie() und barplot() Kreis- und Balkendiagramme

Beispiel:

```
# Nutze dazu ein Tabellenobjekt tab1 von oben  
pie(tab1)  
barplot(tab1)  
?pie; ?barplot
```


Kreis- Balkendiagramme (für 1 qualitatives Merkmal)

pie() und barplot() Kreis- und Balkendiagramme

Beispiel:

```
# Nutze dazu ein Tabellenobjekt tab1 von oben
```

```
pie(tab1)
```

```
barplot(tab1)
```

```
?pie; ?barplot
```

pie(x, labels, col, main)

x: ein Vektor

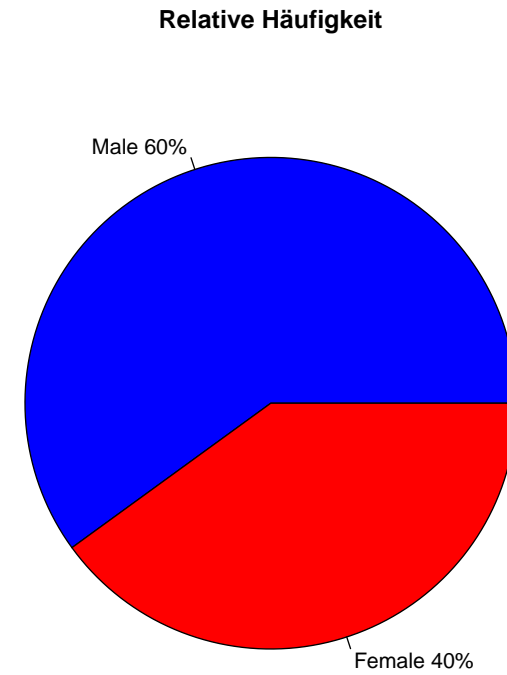
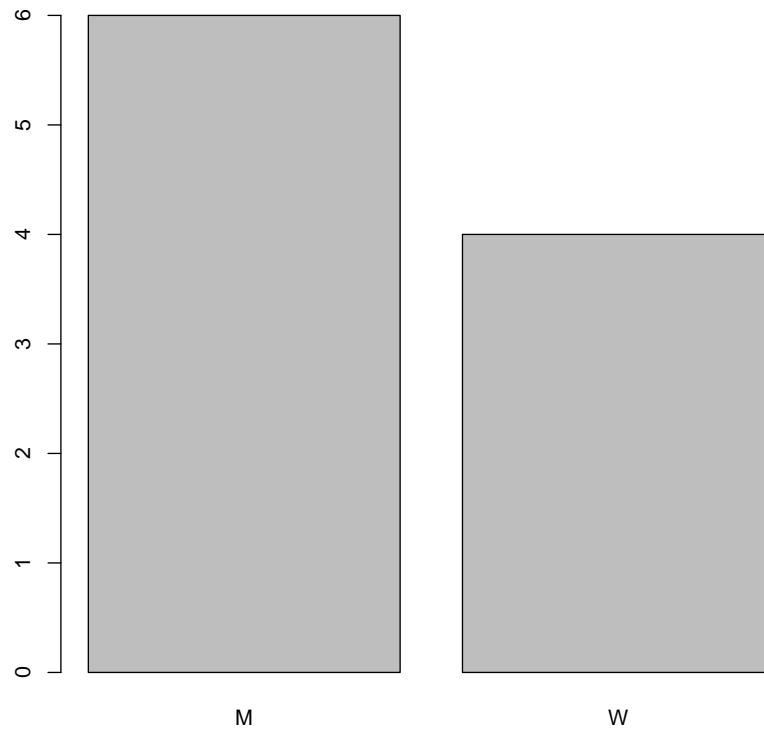
labels: Namen für die Kreisstücke

col: Vector von Farben

main: Titel

```
pie(tab1, labels=c("Male_60%", "Female_40%"),  
     col=c("blue", "red"), main="Relative_Häufigkeit")  
colors() # Namen vordefinierter Farben
```

Kreis- & Balkendiagramme



Übung 3.1

In der Hilfe zu den meisten Funktionen gibt es eine Reihe von Beispielen, die man ausführen kann. Führen Sie die Befehle aus der Hilfe zur Funktion `pie()` aus.

Übung 3.2

Erzeugen Sie wie folgt den künstlichen Datensatz *KlasseB*:

```
set.seed(123)
```

```
KlasseB <- sample(1:6, size = 50, replace = TRUE,  
                  prob = c(.05, .2, .5, .2, .03, .02))
```

KlasseB stelle nun die Notenverteilung in einer Klasse dar.

Erstellen Sie Häufigkeitstabellen für die abs./rel. Häufigkeiten und die kumulativen abs./rel. Häufigkeiten der Notenverteilung.

Stellen Sie die Verteilung der Daten grafisch dar.

Übung 3.2: Lösung

```
tab1 <- table(KlasseB) ; tab1
tab2 <- prop.table(tab1) ; tab2
tab3 <- cumsum(tab1) ; tab3
tab4 <- cumsum(tab2) ; tab4
pie(tab1) ; pie(tab2)
barplot(tab1) ; barplot(tab2)
barplot(tab3) ; barplot(tab4)
```

Übung 3.2: Lösung

```
tab1 <- table(KlasseB) ; tab1
tab2 <- prop.table(tab1) ; tab2
tab3 <- cumsum(tab1) ; tab3
tab4 <- cumsum(tab2) ; tab4
pie(tab1) ; pie(tab2)
barplot(tab1) ; barplot(tab2)
barplot(tab3) ; barplot(tab4)
```

Vorschläge zur Verbesserung der Grafik?
(lese in ?barplot zu names.arg)

Balkendiagramme mit *barplot()*

```
barplot(height, width=1 , space=NULL, names.arg=NULL,  
horiz=FALSE, col, main, , ... )
```

height: *Vector oder Matrix von Werten welche die Balken beschreiben...*

width: *optional, Vektor der Balkenbreiten.*

space: *Menge an Platz (relativ zur durchschnittlichen Balkenbreite), der vor jedem Balken freigehalten wird.*

names.arg: *Vektor von Namen, die unter die Balken geplottet werden.*

horiz: *Bei FALSE werden die Balken vertikal gezeichnet, sonst horizontal.*

xlab/ylab: *Beschriftung für x- bzw. y-Achse.*

Übung 3.3

Modifizieren Sie die Voreinstellungen der Optionen *space*, *names.arg*, *main*, *ylab* für das Erstellen der *barplots*, die Sie für die Lösung der Übung 3.2 verwendet haben, um ansprechendere *barplots* zu erstellen.

Übung zu *barplot()*: Lösung

```
barplot(tab1, main="Notenverteilung", space=1.4,  
        names.arg=paste("Note",1:6),  
        ylab="Absolute_Häufigkeit")  
abline(0,0) # siehe ?abline
```

Übung zu *barplot()*: Lösung

```
barplot(tab1, main="Notenverteilung", space=1.4,  
        names.arg=paste("Note",1:6),  
        ylab="Absolute_Häufigkeit")  
abline(0,0) # siehe ?abline  
  
# oder zuerst main, ylab, names.arg definieren:  
main1 <- "Notenverteilung"  
lab1 <- "Absolute_Häufigkeit"  
name1 <- paste("Note", 1:6, sep="_") # siehe ?paste  
barplot(tab1, main=main1, space=1.4,  
        names.arg=name1, ylab=lab1)  
abline(0,0)
```


Übung zu *barplot()*: Lösung

```
barplot(tab1, main="Notenverteilung", space=1.4,  
        names.arg=paste("Note",1:6),  
        ylab="Absolute_Häufigkeit")  
abline(0,0) # siehe ?abline  
  
# oder zuerst main, ylab, names.arg definieren:  
main1 <- "Notenverteilung"  
lab1 <- "Absolute_Häufigkeit"  
name1 <- paste("Note", 1:6, sep="_") # siehe ?paste  
barplot(tab1, main=main1, space=1.4,  
        names.arg=name1, ylab=lab1)  
abline(0,0)
```

paste("Note",1:6) verknüpft die Argumente "Note" und c(1,2,3,4,5,6) zum Vektor c("Note 1",...,"Note 6").

Mehrere Grafiken in einem Bild

```
op <- par(mfrow = c(1, 2)) # 1 x 2 Bilder auf einem Plot
barplot(table(data1), main="Säulendiagramm")
pie(table(data1),
     labels=c("Male_60%", "Female_40%"),
     col=c("blue", "red"),
     main="Kreisdiagramm")
par(op)
```

1. `op <- par(mfrow = c(1, 2))` teilt die Grafikausgabe in 1 Zeile und 2 Spalten auf.
Die ursprünglichen Grafikeinstellungen werden in die Variable `op` kopiert.
2. `par(op)` setzt die Grafikeinstellungen auf die in `op` gespeicherten Werte zurück.

Kreis- & Balkendiagramme: Beispiel zu *ggplot2*

Alternative Darstellung mit **ggplot2**:

```
# Für ggplot2 Daten in data.frame speichern  
# (Spalten sind die Variables, Zeilen die Beobachtungen).  
df1 <- as.data.frame(tab1)
```

```
library(ggplot2) # Paket laden
```

```
# ggplot-Objekt initialisieren
```

```
p11 <- ggplot(data=df1, aes(x=data1, y=Freq)); p11
```

```
p11 <- p11 + geom_bar(stat="identity"); p11 # Barplot machen
```

```
p12 <- p11 + coord_flip(); p12 # Koordinatensystem spiegeln
```

Plots in **ggplot2** können Schritt für Schritt zusammengebaut werden.

Neue Eigenschaften kann man mit dem "+"-Operator zu einem bestehenden Plot-Objekt hinzufügen.

Kreis- & Balkendiagramme: Beispiel zu *ggplot2*

Um ein Tortendiagramm mit **ggplot2** zu erstellen, kann man einem Balkendiagramm ein Polarkoordinatensystem zuweisen:

```
# Pie-Plot: Ein Barplot mit Polarkoordinatensystem
pie <- ggplot(df1, aes(x="", y=Freq, fill=data1)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start=0); pie
```

Bei der Initialisierung bewirkt **aes(fill=data1)**, dass unterschiedliche Farben für die Tortenstücke der unterschiedlichen Werte in *data1* gewählt werden.

Störende Elemente mit **theme_void()** entfernen. Einen Titel fügt man mit **ggtitle()** hinzu, Beschriftungen mit **geom_text()**:

```
pie2 <- pie +
  theme_void() +
  ggtitle("Gewichtsverteilung:") +
  geom_text(aes(label = Freq/sum(Freq)),
            position = position_stack(vjust = 0.5))

pie2
```

Mehrere Grafiken in einem Bild (*ggplot2*)

Keine Standardfunktionalität. Einfache Möglichkeit mittels `grid.arrange()` aus dem Package `gridExtra`:

```
library(gridExtra)

# p11 und pie nebeneinander in einer Zeile
grid.arrange(p11, pie, nrow = 1)

# p11, p12, pie und pie2 gemeinsam anordnen
grid.arrange(
  grobs = list(pie, pie2, p11, p12),
  widths = c(2, 1, 1),
  layout_matrix = rbind(c(1, 2, NA),
                        c(3, 3, 4))
)
```

Grafiken abspeichern I

Für die Grafikausgabe stehen eine Reihe von *Devices* zur Verfügung: *postscript, pdf, jpeg, bmp, png, emf*.

Möglichkeiten eine Grafiken zu speichern:

a) Man plottet die Grafik (BildschirmAusgabe) und benutzt die Funktionalität der Benutzeroberfläche von R (*Datei* \rightsquigarrow *speichern als*) bzw. RStudio (im Reiter *Plots*: *Export* \rightsquigarrow *Save as Image...* / *Save as PDF...*)

b) Über die entsprechenden Funktionen. Bspw. erzeugt man eine Grafik als PostScript-Datei so:

```
postscript()  
pie(table(data1))  
dev.off()
```

Grafiken abspeichern II

b1) Bevor man die Grafik erstellt, wählt man ein *Device*, z.B. durch:


```
postscript("name.ps") #oder:  
pdf("name.pdf"), .. # relativer Pfad #oder:  
pdf("D:/.../name.pdf") # absoluter Pfad
```

b2) dann wird die Grafik ins Device geplottet:

```
pie(table(data1),  
     labels=c("Male_60%", "Female_40%"),  
     col=c("blue", "red"),  
     main="Relative_Häufigkeit")
```

b3) dann wird das *Device* geschlossen:

```
dev.off()
```

Bemerkung: schreibt man in b1) nur **postscript()** bzw. **pdf()** usw., so speichert  die Grafik im Arbeitsverzeichnis unter dem Namen *Rplot*.

Für ggplot2: siehe **?ggsave**

Übung 3.4

Laden Sie den Datensatz *bakterien.txt* und erstellen Sie eine Tabelle der absoluten/relativen Häufigkeiten für die qualitativen Merkmale **resistenz** und **farbe**.

Stellen Sie Ihre Ergebnisse mit Hilfe von Kreis- und Balkendiagrammen dar.

Hinweis: benutzen Sie `table()` für die einzelnen Variablen!

Übung 3.5

Erstellen Sie ein Bild, das die 4 Grafiken, die Sie für die Übung 3.4 erstellt haben, in einem gemeinsamen Plot enthält.

Speichern Sie das Bild in einem von Ihnen gewählten Verzeichnis.

Übung 3.4: Lösung

```
myfile1 <- "bakterien.txt" # ggf. Pfad absolut angeben
bakterien <- read.table(file=myfile1, h=T)
head(bakterien)
attach(bakterien)
x <- table(resistenz); x
y <- table(farbe); y
x1 <- prop.table(x); x1
y1 <- prop.table(y); y1
pie(x, main="Bakterienresistenz")
barplot(y, main="Bakterienfarbe")
detach(bakterien)
```

Übung 3.5: Lösung

```
myfile0 <- "name1.pdf"
attach(bakterien)
pdf(myfile1)
op <- par(mfrow = c(2, 2)) # 2 x 2 Grafiken in einem Plot
pie(x, main="Bakterienresistenz")
pie(y, main="Bakterienfarbe")
barplot(x, main="Bakterienresistenz")
barplot(y, main="Bakterienfarbe")
par(op)
dev.off()
detach(bakterien)
```

Übung 3.6

Erzeugen Sie *Wetter2017* mittels:

```
set.seed(2017)
Wetter2017 <- sample(c("kalt", "mild", "warm", "heiss"),
                     size = 365, replace = T,
                     prob = c(0.5, 0.35, 0.12, 0.03))
```

Erstellen Sie die abs./rel. Häufigkeiten und kumulative abs./rel. Häufigkeiten für diesen Daten.

Stellen Sie die Verteilung der Daten grafisch dar.

Kreis- und Balkendiagramme für ordinal-skalierte Merkmale

```
x <- table(Wetter2017); x  
heiss kalt mild warm  
11 193 120 41
```

```
prop.table(x)  
heiss kalt mild warm  
0.030 0.529 0.329 0.112
```

Die Daten sind nicht nach Intensität geordnet!

```
pie(table(Wetter2017))  
barplot(table(Wetter2017))
```

Auch hier sieht man die falsche Anordnung der Säulen!

Kreis- und Balkendiagramme für ordinal-skalierte Merkmale

Die Variable *Wetter2017* hat den Datentyp *character*. Wir erzeugen eine neue Variable, die den Datentyp *factor* hat und definieren eine neue Ordnung für die Daten. Dies kann man mit dem Befehl *factor()* oder *ordered()* und Anwendung der Option *levels* erfolgen:

```
wlevels <- c("kalt", "mild", "warm", "heiss")
Wetter2017.ord <- ordered(Wetter2017,
                          levels=wlevels) # bzw.:
Wetter2017.fac <- factor(Wetter2017,
                         levels=wlevels)

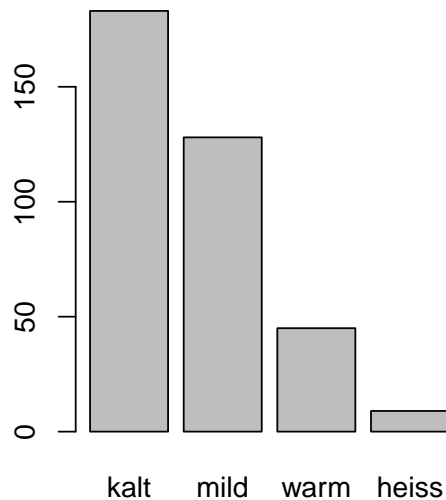
x <- table(Wetter2017.ord); x
kalt mild warm heiss
193  120  41  11

prop.table(x)
kalt mild warm heiss
0.529 0.329 0.112 0.030
```

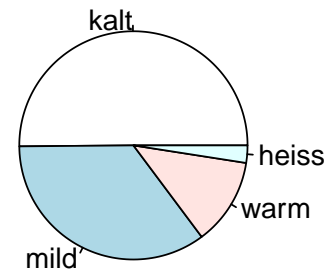
Kreis- und Balkendiagramme für ordinal-skalierte Merkmale

```
x <- table(Wetter2017.ord)
op <- par(mfrow = c(1, 2))
barplot(x, main="Säulendiagramm")
abline(0, 0)
pie(x, main="Kreisdiagramm")
par(op)
```

Säulendiagramm



Kreisdiagramm



Übung 3.7

Installieren Sie das Paket *mixsmsn* (aus dem offiziellen CRAN-Repository).

- a. Laden Sie den Datensatz *bmi* aus diesem Paket und lesen Sie die Hilfefunktion zu diesem Datensatz.
- b. Fügen Sie dem Datensatz eine neue Spalte *BMI* mit
BMI= "Low" für $\text{bmi} < 18.5$,
BMI= "Normal" für $18.5 \leq \text{bmi} \leq 25$ und
BMI= "High" für $\text{bmi} > 25$.
- c. Stellen Sie die abs.- und rel. Häufigkeiten von *BMI* mit Tabellen und Grafiken dar.

Lösung

```
# Teil a.
install.packages("mixsmsn")
data(bmi, package = "mixsmsn")

# Teil b.
bmi$BMI <- ifelse(bmi$bmi<18.5, "Low",
                 ifelse(bmi$bmi>=18.5 & bmi$bmi<=25,
                        "Normal", "High"))

# Teil c.
attach(bmi) # Objekte aus bmi verfügbar machen
BMI.ord <- ordered(BMI, levels=c("Low", "Normal", "High"))
x <- table(BMI.ord); x
y <- prop.table(x); y
barplot(x, main="BMI-Verteilung", ylab="abs. Häufigkeit")
abline(0,0)
barplot(y, main="BMI-Verteilung", ylab="rel. Häufigkeit")
abline(0,0)
detach(bmi) # attach rückgängig machen
```


Charakteristische Maßzahlen (für 1 qualitatives Merkmal)

Modalwert kann man für **nominal-skalierte** Merkmale bilden.

```
data1 <- rep(c("leicht", "normal", "schwer"), c(16, 40, 4))  
which.max(table(data1))  
normal
```

```
set.seed(123)  
KlasseB <- sample(c(1, 2, 3, 4, 5), size=200, replace=T,  
                  prob=c(0.1, 0.4, 0.3, 0.15, 0.05))  
which.max(table(KlasseB))  
2
```

Charakteristische Maßzahlen (für 1 qualitatives Merkmal)

Median und p -Quantile kann man für **ordinal-skalierte** Merkmale bilden.

```
quantile(KlasseB, probs=0.5)
```

```
3
```

```
?quantile
```

```
quantile(x, probs=seq(0, 1, 0.25), type=7, na.rm=FALSE)
```

```
0% 25% 50% 75% 100%
```

```
74.0 476.0 878.0 1016.5 1155.0
```

```
y <- c(1, 2, 4, 4, 5, 6, 7, 8)
```

```
quantile(y, probs=0.5, type=1)
```

```
50%
```

```
4
```

```
quantile(y, probs=0.5, type=2)
```

```
50%
```

```
4.5
```

3. Deskriptive Statistik für:

1. ein qualitatives Merkmal (univariat)
2. **zwei qualitative Merkmale (bivariat)**
3. ein quantitatives Merkmal (univariat)
4. ein qualitatives und ein quantitatives Merkmal (bivariat)
5. zwei quantitative Merkmale (bivariat)

Kreuztabelle/Kontingenztafel für zwei qualitative Merkmale

Die Ergebnisse von `table` und `prop.table` sind *arrays*!

```
class(tab1); class(prop.table(tab1))  
array
```

Übung 3.8

Laden Sie den Datensatz *HairEyeColor* aus dem Paket *datasets* und lesen Sie die Hilfe zu diesem Datensatz:

```
data(HairEyeColor, package = "datasets")
```

a) Führen Sie Folgendes aus und interpretieren Sie:

```
dim(HairEyeColor)  
mtab <- HairEyeColor[, ,1]  
wtab <- HairEyeColor[, ,2]  
tab1 <- HairEyeColor[1, ,]  
tab2 <- HairEyeColor[2, ,]  
tab3 <- HairEyeColor[, 1, ]  
tab4 <- HairEyeColor[, 3, ]  
tab5 <- HairEyeColor[1, ,1]
```

Kreuztabelle: Übung

b) Stellen Sie die Verteilungen der folgenden Merkmale durch Kreisdiagramme dar:

- Haarfarbe von Männern mit braunen Augen
- Augenfarbe schwarzhaariger Frauen
- Geschlecht von Personen mit braunen Augen und schwarz Haaren

Passen Sie die Farbe der Kreissektoren den Haar- und Augenfarben an.

Kreuztabelle: Lösung

```
t1 <- HairEyeColor; t1
col1<-c("black","brown","red","gold")
col2<-c("brown","blue","khaki3","green")
col3<-c("blue","red")

t1[, 1,1] # Braunäugige Männer
pie(t1[, 1,1],col=col1,main="Hair_Color_distribution",
    sub="for_males_with_brown_eyes")

t1[1, ,2] # Schwarzhaarige Frauen
pie(t1[1, ,2],col=col2,main="Eye_Color_distribution",
    sub="for_females_with_black_Hair")

t1[1,1 ,] # Schwarzhaarig und brauäugig
pie(t1[1,1 ,],col=col3 ,main="Sex_distribution",
    sub="for_black_haired_people_with_brown_eyes")
```

Tabellen manipulieren: *ftable()* , *structable()*

Die Funktionen *structable()* (enthalten im Paket *vcd*) und *ftable()* sind geeignet für Kontingenztafeln mit mehr als 2 Merkmalen.

```
t1 <- HairEyeColor; t1; dim(t1)
```

```
ftab1<-ftable(t1) ; ftab1
```

```
# installiere und lade das vcd-Paket aus dem Netz!
```

```
install.packages("vcd")
```

```
library(vcd)
```

```
ktab1 <- structable(t1)
```

```
ktab2 <- structable(formula=Hair~Sex+Eye, data=t1)
```

```
# Formula: definiert col ~ row Variablen.
```

```
ktab3 <- structable(formula=Hair+Sex~Eye, data=t1) # usw.
```

Übung: Tabellen und Plots für 2 qualitative Merkmale

Übung 3.9

Zuerst generieren wir unsere eigene *hair-eye-sex*-Datei.

```
x <- sample(c("M", "W"),
            size=1000, replace=T, prob=c(0.35, 0.65))
y <- sample(c("black", "brown", "red", "blond"),
            size=1000, replace=T, prob=c(0.2, 0.4, 0.1, 0.3))
z <- sample(c("blue", "brown", "hazel", "green"),
            size=1000, replace=T, prob=c(0.3, 0.4, 0.2, 0.1))
gha <- as.data.frame(cbind(x, y, z))
colnames(gha) <- c("Geschlecht", "Haarfarbe", "Augenfarbe")
```

Wir wollen für diesen Datensatz alle mögliche 1-, 2- und 3-dimensionalen Tabellen und Grafiken erstellen.

```
# Farben für Merkmalsausprägungen:
col1 <- c("black", "brown", "red", "gold")
col2 <- c("brown", "blue", "khaki3", "green")
col3 <- c("blue", "red")
attach(gha) # Variablen von gha verfügbar machen
```


	Geschlecht	Haarfarbe	Augenfarbe
1	W	red	brown
2	M	blond	brown
3	M	brown	blue
4	W	black	green
5	W	brown	hazel
6	W	blond	blue
7	W	black	brown
8	W	blond	blue
9	W	black	blue
10	W	blond	hazel
11	M	brown	brown
12	M	brown	hazel
.	.	.	.

Übung: Tabellen und Plots für 2 qualitative Merkmale

1-dimensionale Tabellen/Grafiken:

```
tab1 <- table(Geschlecht); tab1  
barplot(tab1)  
tab1 <- prop.table(tab1)  
pie(tab1, labels=c("Male", "Female"),  
     col=col3, main="Sex")
```

Übung: Tabellen und Plots für 2 qualitative Merkmale

1-dimensionale Tabellen/Grafiken:

```
tab1 <- table(Geschlecht); tab1  
barplot(tab1)  
tab1 <- prop.table(tab1)  
pie(tab1, labels=c("Male", "Female"),  
     col=col3, main="Sex")
```

```
tab2 <- table(Haarfarbe); tab2 # Umordnen:  
tab2 <- tab2[c(1,3,4,2)]; tab2  
tab2 <- prop.table(tab2); tab2  
pie(tab2, main="Haarfarbe", col=col1)  
barplot(tab2, main="Haarfarbe", col=col1)
```

Übung: Tabellen und Plots für 2 qualitative Merkmale

1-dimensionale Tabellen/Grafiken:

```
tab1 <- table(Geschlecht); tab1
barplot(tab1)
tab1 <- prop.table(tab1)
pie(tab1, labels=c("Male", "Female"),
     col=col3, main="Sex")
```

```
tab2 <- table(Haarfarbe); tab2 # Umordnen:
tab2 <- tab2[c(1,3,4,2)]; tab2
tab2 <- prop.table(tab2); tab2
pie(tab2, main="Haarfarbe", col=col1)
barplot(tab2, main="Haarfarbe", col=col1)
```

```
tab3 <- table(Augenfarbe); tab3 <- tab3[c(2,1,4,3)]
tab3 <- prop.table(tab3)
pie(tab3, main="Augenfarbe", col=col2)
barplot(tab3, main="Augenfarbe", col=col2)
```

Übung: Tabellen und Plots für 2 qualitative Merkmale

1-dimensionale Tabellen/Grafiken:

```
tab1 <- table(Geschlecht); tab1
barplot(tab1)
tab1 <- prop.table(tab1)
pie(tab1, labels=c("Male", "Female"),
     col=col3, main="Sex")
```

```
tab2 <- table(Haarfarbe); tab2 # Umordnen:
tab2 <- tab2[c(1,3,4,2)]; tab2
tab2 <- prop.table(tab2); tab2
pie(tab2, main="Haarfarbe", col=col1)
barplot(tab2, main="Haarfarbe", col=col1)
```

```
tab3 <- table(Augenfarbe); tab3 <- tab3[c(2,1,4,3)]
tab3 <- prop.table(tab3)
pie(tab3, main="Augenfarbe", col=col2)
barplot(tab3, main="Augenfarbe", col=col2)
```

```
graphics.off()
```

Übung: Tabellen und Plots für 2 qualitative Merkmale

2-dimensionale Tabellen

```
tab4 <- table(Augenfarbe, Haarfarbe); tab4
```

	black	blond	brown	red
blue	68	75	123	28
brown	93	102	152	43
green	25	40	39	9
hazel	43	60	77	23

Übung: Tabellen und Plots für 2 qualitative Merkmale

2-dimensionale Tabellen

```
tab4 <- table(Augenfarbe, Haarfarbe); tab4
```

	black	blond	brown	red
blue	68	75	123	28
brown	93	102	152	43
green	25	40	39	9
hazel	43	60	77	23

```
# tabx <- table(Augenfarbe, Geschlecht); tabx
tab5 <- prop.table(tab4); tab5
tab5.0 <- prop.table(tab4, margin=1); tab5.0
tab5.1 <- prop.table(tab4, margin=2); tab5.1
tab6 <- addmargins(tab4); tab6
tab7 <- addmargins(tab5); tab7
tab7.1 <- addmargins(tab5.1, margin=1); tab7.1
```

3-dimensionale Tabellen

```
taby <- table(gha); taby  
ftaby <- ftable(taby); ftaby  
ktaby1 <- structable(taby) ; ktaby1  
ktaby2 <- structable(Augenfarbe+Geschlecht~Haarfarbe,taby)  
ktaby2  
detach(gha)
```


Übung 3.10

Generieren Sie mit dem folgenden R-Skript den neuen Datensatz *gha1*.

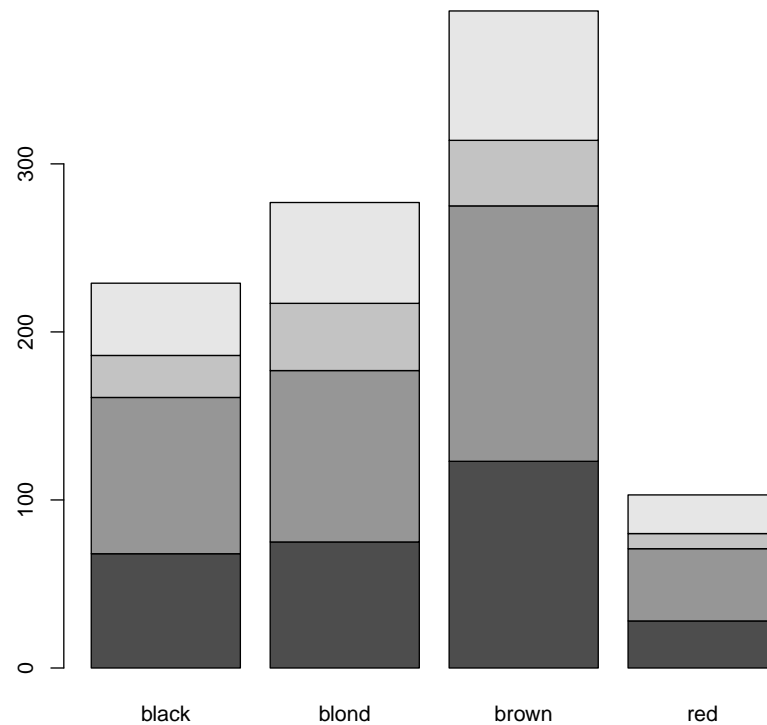
Vergleichen Sie diesen Datensatz mit dem Datensatz *gha* im Hinblick auf die Simulationsmethode.

Erstellen Sie Tabellen und Grafiken für diesen Datensatz.

```
x <- sample(c("M", "W"), size=10000,
            replace=T, prob=c(0.3, 0.7))
y <- ifelse(x=="M",
            sample(c("black", "brown", "red", "blond"),
                  size=length(x=="M"),
                  prob=c(0.4, 0.1, 0.2, 0.3), replace=T),
            sample(c("black", "brown", "red", "blond"),
                  size=length(x=="W"),
                  prob=c(0.2, 0.6, 0.1, 0.1), replace=T))
gha1 <- data.frame(Geschlecht=x, Haarfarbe=y)
```

Gruppierte Balkendiagramme für 2 qualitative Merkmale

```
attach(gha)
tab4 <- table(Augenfarbe, Haarfarbe); tab4
barplot(tab4, beside=F)
```



Gruppierte Balkendiagramme: *barplot()*

```
barplot(height,..., beside = FALSE, horiz = FALSE, legend = TRUE,  
col,...)
```

height: Vektor oder Matrix... Falls **Matrix** und *beside* == *FALSE*, dann bezieht sich jeder Balken des Plots auf eine Spalte der Matrix, wobei die Werte in der Spalte die Höhen von gestapelten Balkenabschnitten angeben...

beside=FALSE: .. bei *TRUE* werden die Spalten der Matrix als nebeneinander als gruppierte Balken dargestellt.

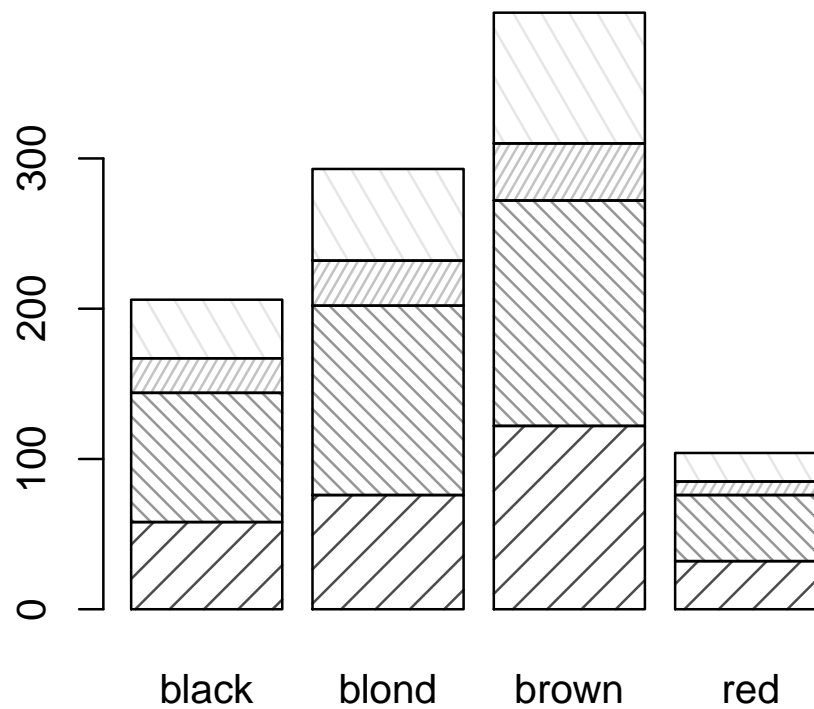
density = NULL: Vektor für die Dichte der Schattierungslinien in Zeilen pro Inch (für die Balken und Balkenkomponenten).

angle = 45: Die Steigung der Schattenlinien.

legend = TRUE/FALSE: Legende plotten - ja oder nein.

Gruppierte Balkendiagramme für 2 qualitative Merkmale

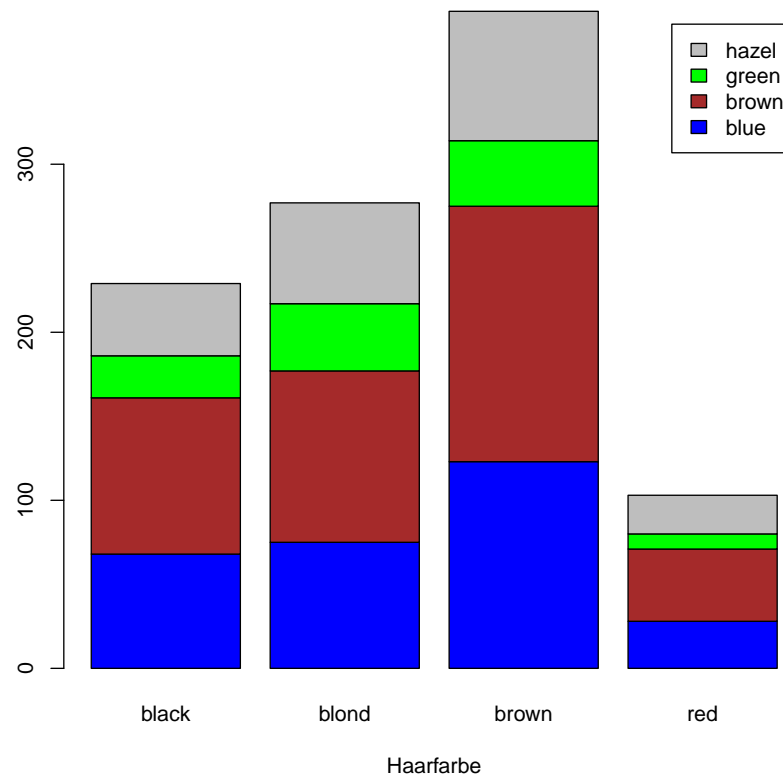
```
barplot(tab4, beside=FALSE,  
        density=c(10,20,30,10),  
        angle=c(45,135,60,120))
```



Gruppierte Balkendiagramme für 2 qualitative Merkmale

Übung 3.11

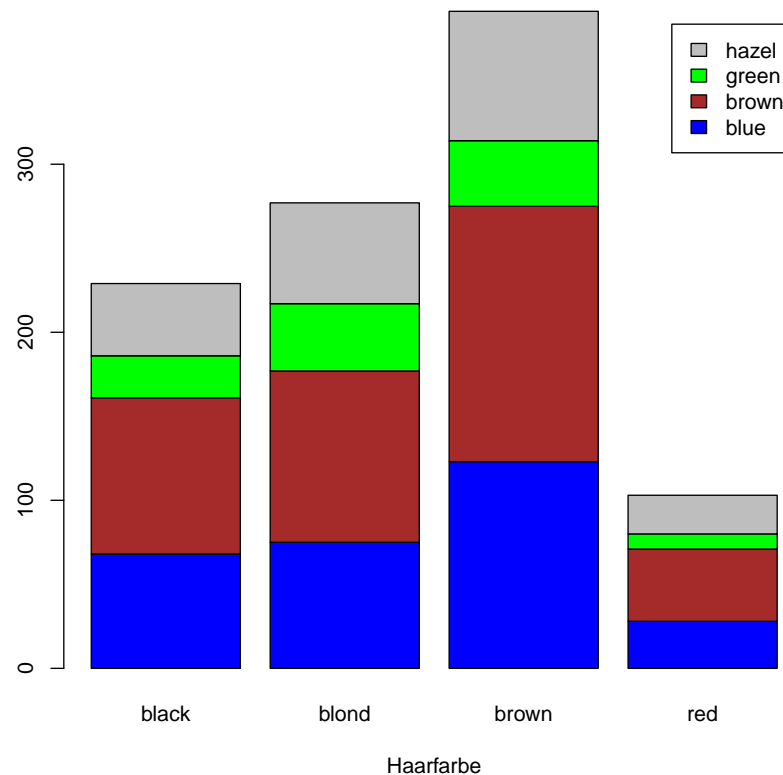
Versuchen Sie das *Balkendiagramm* der letzten Seite mit passenden Farben darzustellen.



Gruppierte Balkendiagramme für 2 qualitative Merkmale

Übung 3.11

Versuchen Sie das *Balkendiagramm* der letzten Seite mit passenden Farben darzustellen.

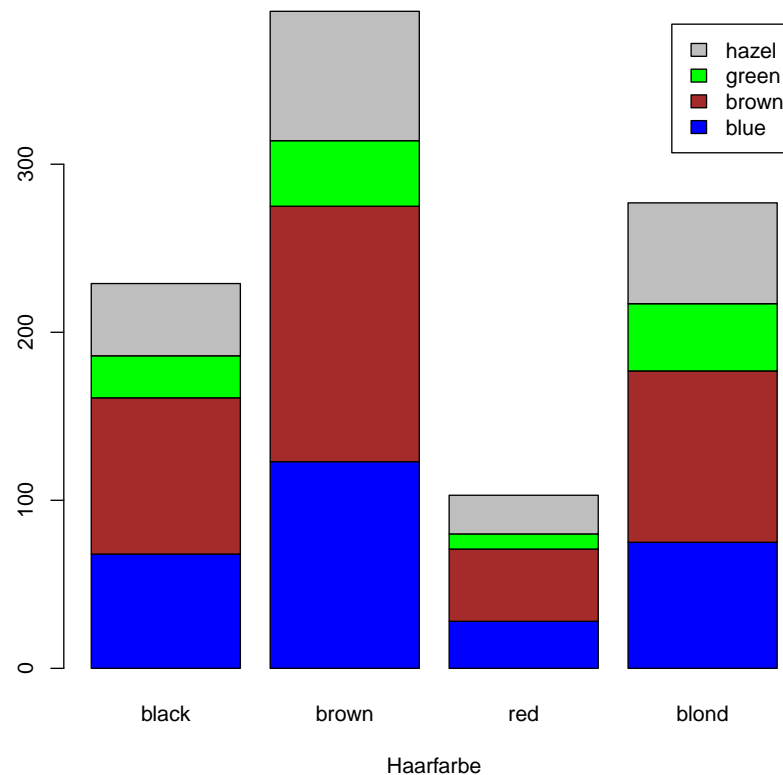


```
barplot(tab4, beside=FALSE, legend=TRUE,  
        col = c("blue", "brown", "green", "khaki3"),  
        xlab="Haarfarbe")
```

Gruppierte Balkendiagramme für 2 qualitative Merkmale

Übung 3.12

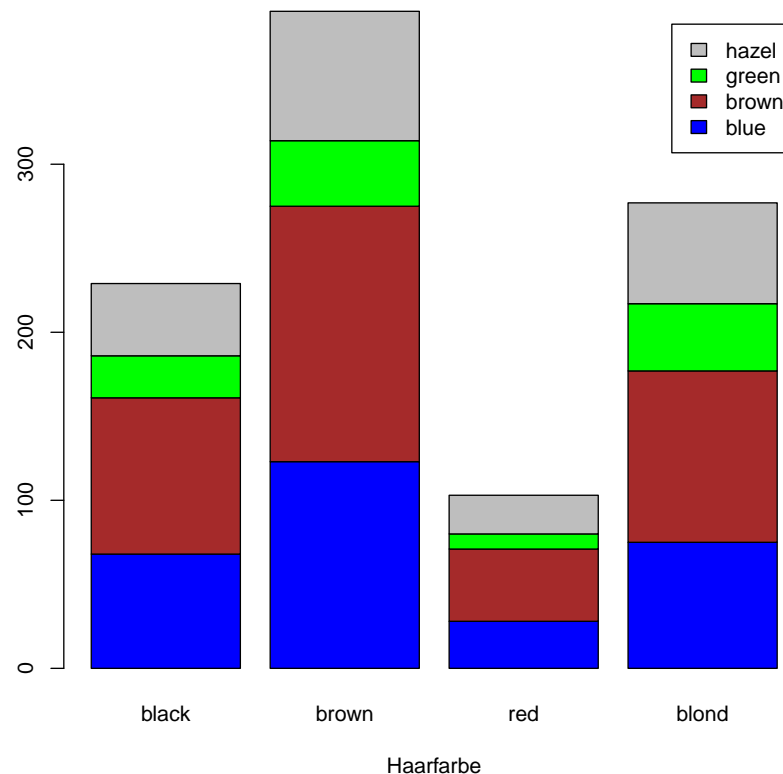
Jetzt versuchen Sie das Merkmal *Haarfarbe* in einer geordneten Reihe darzustellen.



Gruppierte Balkendiagramme für 2 qualitative Merkmale

Übung 3.12

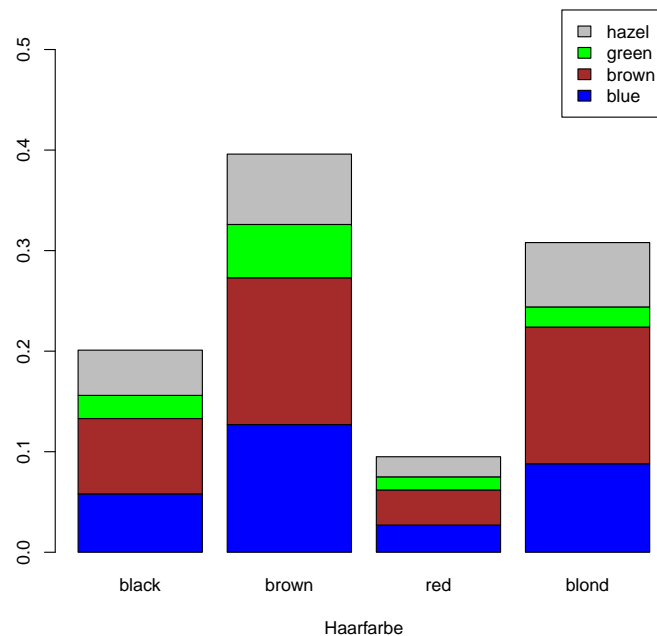
Jetzt versuchen Sie das Merkmal *Haarfarbe* in einer geordneten Reihe darzustellen.



```
tab5 <- tab4[,c(1,3,4,2)]  
barplot(tab5, beside=FALSE, legend=TRUE,  
        col = c("blue", "brown", "green", "khaki3"),  
        xlab="Haarfarbe")
```


Gruppierte Balkendiagramme für 2 qualitative Merkmale

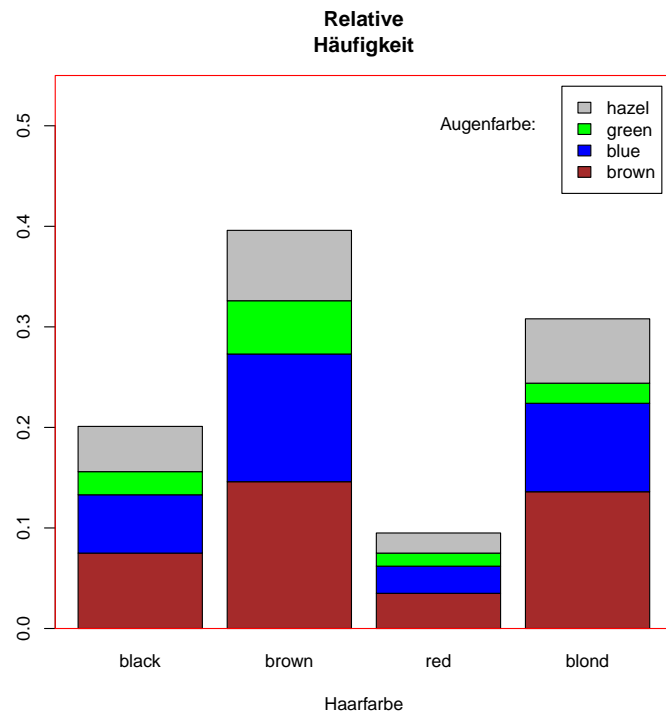
```
tab5 <- prop.table(tab5)
barplot(tab5, beside=FALSE, legend=TRUE,
        col = c("blue", "brown", "green", "khaki3"),
        xlab="Haarfarbe", ylim=c(0, .65))
```



Versuchen Sie es jetzt mit `tab6 <- tab4[c(2,1,3,4),c(1,3,4,2)]`

Gruppierte Balkendiagramme für 2 qualitative Merkmale

```
tab6 <- tab4[c(2,1,3,4),c(1,3,4,2)]; tab6 <- prop.table(tab6)
barplot(tab6, beside=FALSE, legend=TRUE,
        col = c("brown", "blue", "green", "khaki3"),
        xlab="Haarfarbe", ylim=c(0,0.55),
        main="Relative_Häufigkeit")
text(3.5,0.5,label="Augenfarbe:")
box(col="red")
```



Kreuztabellen (für 2 qualitative Merkmale) I

Beispiel: Absolute Häufigkeit der Blutgruppe (*ABO*) nach Geschlecht

Geschl.	Blutgr.	A	B	AB	O	Kumuliert
männlich		43	12	7	38	100
weiblich		40	16	9	35	100
Alle		83	28	16	73	200

Kreuztabellen (für 2 qualitative Merkmale) II

Betrachte 2 nominalskalierte Merkmale S und T mit Ausprägungen:

$S : S_1, S_2, \dots, S_I$

$T : T_1, T_2, \dots, T_J$

Beispiel: Merkmale Geschlecht (S) und Blutgruppe (T) mit Ausprägungen:

Geschlecht: w, m

Blutgruppe: A, B, AB, O

Es gibt insgesamt 8 mögliche Ausprägungen:

(S_i, T_j) , für $i = 1, 2, j = 1, 2, 3, 4$

n_{ij} : die Anzahl der Merkmale mit Ausprägung S_i und T_j

Kreuztabellen (für 2 qualitative Merkmale) III

S \ T	T_1	T_2	\dots	T_J	Σ
S_1	n_{11}	n_{12}	\dots	n_{1J}	$n_{1.}$
S_2	n_{21}	n_{22}	\dots	n_{2J}	$n_{2.}$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
S_I	n_{I1}	n_{I2}	\dots	n_{IJ}	$n_{I.}$
Σ	$n_{.1}$	$n_{.2}$	\dots	$n_{.J}$	n

Randhäufigkeiten:

$$n_{i.} = \sum_{j=1}^J n_{ij}$$

$$n_{.j} = \sum_{i=1}^I n_{ij}$$

Relative und Rand-/Marginalhäufigkeiten

$h_{T_j|S_i} = \frac{n_{ij}}{n_{i.}} =$ **Bedingte** relative Häufigkeit von T_j unter der Bedingung, dass die Ausprägung S_i gegeben ist.

Die Summe der Zahlen in jeder Zeile der Tabelle ist gleich 1.

$h_{S_i|T_j} = \frac{n_{ij}}{n_{.j}} =$ **Bedingte** relative Häufigkeit von S_i unter der Bedingung, dass die Ausprägung T_j gegeben ist.

Die Summe der Zahlen in jeder Spalte der Tabelle ist gleich 1.

$h_{ij} = \frac{n_{ij}}{n} =$ Relative Häufigkeit der Einheiten mit den Ausprägungen S_i und T_j .

Beispiel: Kreuztabellen lesen

Absolute Häufigkeit der Blutgruppe (*ABO*) nach Geschlecht

	A	B	AB	O	Kumuliert
männlich	43	12	7	38	100
weiblich	40	16	9	35	100
Alle	83	28	16	73	200

Relative Häufigkeit der Blutgruppe (*ABO*) nach Geschlecht

	A	B	AB	O	Kumuliert
männlich	0.430	0.120	0.070	0.380	1
weiblich	0.400	0.160	0.090	0.350	1
Alle	0.415	0.140	0.080	0.365	1

Erläuterung: 43% der Männer haben Blutgruppe AB sind weiblich.

Beispiel: Kreuztabellen lesen

Absolute Häufigkeit der Blutgruppe A

	A	B	AB	O	Kumuliert
männlich	43	12	7	38	100
weiblich	40	16	9	35	100
Alle	83	28	16	73	200

Relative Häufigkeit des Geschlechts nach Blutgruppe (ABO)

	A	B	AB	O	Kumuliert
männlich	0.52	0.43	0.44	0.52	0.5
weiblich	0.48	0.57	0.56	0.48	0.5
Alle	1	1	1	1	1

Erläuterung: 56% der Blutgruppe AB sind weiblich.

Beispiel: Kreuztabellen lesen

Absolute Häufigkeit der Blutgruppe (*ABO*) nach Geschlecht in einem Datensatz

	A	B	AB	O	Kumuliert
männlich	43	12	7	38	100
weiblich	40	16	9	35	100
Alle	83	28	16	73	200

Relative Häufigkeit der Blutgruppe (*ABO*) und Geschlecht

	A	B	AB	O	Kumuliert
männlich	0.215	0.060	0.035	0.190	0.5
weiblich	0.200	0.080	0.045	0.175	0.5
Alle	0.415	0.14	0.08	0.365	1

Erläuterung: 8% aller Patienten haben Blutgruppe B und sind weiblich.

```
x <- sample(c("M", "W"),  
            size=100,  
            replace=TRUE,  
            prob=c(0.25, 0.75)) # siehe ?sample
```

Mit *sample(x, size, replace = FALSE, prob = NULL)* zieht man zufällig Stichproben aus einem Vektor.

replace = TRUE/FALSE: mit/ohne Zurücklegen

size = 100: es werden 100 Elemente gezogen

prop: jeweilige Wahrscheinlichkeiten

Hier: $X_1, \dots, X_{100} \sim \text{bern}(0.25)$; wobei :

$$P(X_i = M) = 0.25$$

ohne Marginal/Rand, abs. Häufigkeiten

```
x <- sample(c("M", "W"), size=100, replace=T,  
            prob=c(0.25, 0.75)) # siehe ?sample  
y <- sample(c("jung", "alt"), size=100, replace=T,  
            prob=c(0.4, 0.6))  
z <- as.data.frame(cbind(x, y))  
tab1 <- table(z) ; tab1
```

	alt	jung
M	13	6
W	45	36

```
prop.table(tab1)
```

	alt	jung
M	0.13	0.06
W	0.45	0.36

mit Marginal/Rand, abs. Häufigkeiten

```
tab1
```

	alt	jung
M	13	6
W	45	36

```
Summe2<-margin.table(tab1, margin=2) ; Summe2  
58  42
```

```
tab.neu <- rbind(tab1, Summe2); tab.neu # oder  
tab.neu <- rbind(tab1, margin.table(tab1, margin=2))
```

	alt	jung
M	13	6
W	45	36
Summe2	58	42

```
# oder:
```

```
addmargins(tab1, margin=1)
```

mit Marginal/Rand, abs. Häufigkeiten

```
tab1
```

	alt	jung
M	13	6
W	45	36

```
Summe1 <- margin.table(tab1,1) ; Summe1  
19 81
```

```
tab.new <- cbind(tab1, Summe1); tab.new# oder  
tab.new <- cbind(tab1, margin.table(tab1, margin=1))
```

	alt	jung	Summe1
M	13	6	19
W	45	36	81

```
# oder:
```

```
addmargins(tab1,margin=2)
```

mit Marginal/Rand, abs. Häufigkeiten

```
tab1
```

```
      alt  jung  
M     13    6  
W     45   36
```

```
addmargins(tab1) # oder:  
# addmargins(tab1, margin=c(1,2))
```

```
      alt  jung  Sum  
M     13    6   19  
W     45   36   81  
Sum   58   42  100
```

mit Marginal/Rand, abs. Häufigkeiten

```
addmargins(table(z)) # oder:
```

```
addmargins(table(z), margin=c(1, 2))
```

```
addmargins(table(z), margin=1)
```

```
addmargins(table(z), margin=2)
```

ohne Marginal/Rand, rel. Häufigkeiten

```
prop0 <- prop.table(tab1, margin=NULL); prop0 # default
```

	alt	jung
M	0.13	0.06
W	0.45	0.36

```
prop1 <- prop.table(tab1, margin=1); prop1 # Zeilenweise
```

	alt	jung
M	0.684	0.316
W	0.556	0.444

```
prop2 <- prop.table(tab1, margin=2); prop2 # Spaltenweise
```

	alt	jung
M	0.224	0.143
W	0.776	0.857

mit Marginal/Rand, rel. Häufigkeiten

```
tab3 <- margin.table(prop1, margin=1)
tab.new1 <- cbind(prop1, tab3) ; tab.new1
```

	alt	jung	Summe
M	0.684	0.316	1
W	0.556	0.444	1

```
tab4 <- margin.table(prop2, margin=2)
tab.neu1 <- rbind(prop2, tab4) ; tab.neu1
```

	alt	jung
M	0.224	0.143
W	0.776	0.857
Summe	1	1

mit Marginal/Rand, rel. Häufigkeiten

```
addmargins(prop.table(table(z),margin=NULL)) # oder
```

```
addmargins(prop.table(table(z),margin=NULL), margin=c(1,2))
```

```
addmargins(prop.table(table(z),margin=NULL),margin=1)
```

```
addmargins(prop.table(table(z),margin=NULL),margin=2)
```

```
addmargins(prop.table(table(z),margin=1),margin=2)
```

```
addmargins(prop.table(table(z),margin=2),margin=1)
```

Kreuztabellen: Zusammenfassung

```
# einfach - ohne Marginal und abs. Häufigkeiten:  
table(x) # x ein data.frame oder:  
table(x1, x2,...) # x1 x2 Vektoren (Spalten eines Datensatz)  
# x, x1, x2: qualitative Merkmale  
  
# ohne Marginal mit rel. Häufigkeiten:  
prop.table(table(...), margin) # margin = NULL, 1, 2, ..  
  
# mit Marginal mit abs. Häufigkeiten:  
addmargins(table(...), margin)  
  
# mit Marginal mit rel. Häufigkeiten:  
addmargins(prop.table(table(...), margin), margin)
```

Kreuztabellen manipulieren

```
aperm(x, perm=c(2, 1))
```

Übung 3.13

Vertauschen Sie die Dimensionen der Kreuztabellen, die wir für den Data.frame z (Geschlecht und Alter) erstellt haben:

	M	W
alt		
jung		

Übung 3.14

1. Fügen Sie dem Datensatz *bmi* aus der Übung 3.7 mit den Variablen *bmi* und *BMI* zufällig eine weitere Variable *Geschlecht* hinzu (mittels `sample(..)`). Nehmen Sie an, dass $P("M") = 0.4$.
2. Erstellen Sie alle Kreuztabellen (abs./rel. Häufigkeiten ohne/mit Marginal) für die qualitative Variablen.

Lösung

```
library(mixsmsn); data(bmi)
bmi$BMI <- ifelse(bmi$bmi<18.5, "Low",
                 ifelse(bmi$bmi<=25, "Normal", "High"))
bmi$BMI <- ordered(bmi$BMI,
                  levels=c("Low", "Normal", "High"))
x <- sample(c("M", "W"), size=nrow(bmi),
           replace=T, prob=c(0.4, 0.6))
z <- cbind(bmi, x)
tab1 <- table(z[,2], z[,3]); tab1

addmargins(tab1)
addmargins(tab1, margin=1)
addmargins(tab1, margin=2)

prop0 <- prop.table(tab1); prop0
prop1 <- prop.table(tab1, margin=1); prop1
prop2 <- prop.table(tab1, margin=2); prop2
addmargins(prop.table(tab1, margin=NULL))
addmargins(prop.table(tab1, margin=NULL), margin=1)
addmargins(prop.table(tab1, margin=NULL), margin=2)
addmargins(prop.table(tab1, margin=1), margin=2)
addmargins(prop.table(tab1, margin=2), margin=1)
```

3. Deskriptive Statistik für:

1. ein qualitatives Merkmal (univariat)
2. zwei qualitative Merkmale (bivariat)
3. **ein quantitatives Merkmal (univariat)**
4. ein qualitatives und ein quantitatives Merkmal (bivariat)
5. zwei quantitative Merkmale (bivariat)

„Häufigkeitstabellen“ & Histogramme für quantitative Daten

Beispiel 3.1

Krankenhausdaten mit *Bilirubin*-Werten von $n = 12809$ Patienten mit folgenden Merkmalen:

Alter: metrisch, diskret

Geschlecht: nominal

Wert: metrisch, stetig (diskret)

Importiere *bilirubin*-Datei:

```
bilirubin <- read.table(file="Daten/bilirubin.txt",  
                        head=TRUE, sep=";")
```

	ALTER	SEX	Wert
1	56	W	0.00
2	81	W	0.03
3	66	W	0.03
4	59	M	0.04
.	.	.	.

Übung 3.15

Erstellen Sie für die Merkmale *ALTER*, *SEX* und *Wert* (aus den *Bilirubin-Daten*) Häufigkeitstabellen, Kreis- und Balkendiagramme.

Übung 3.15

Erstellen Sie für die Merkmale *ALTER*, *SEX* und *Wert* (aus den *Bilirubin-Daten*) Häufigkeitstabellen, Kreis- und Balkendiagramme.

```
str(bilirubin)
```

```
table(bilirubin[,1])
```

```
table(bilirubin[,2])
```

```
table(bilirubin[,3])
```

```
pie(table(bilirubin[,1]))
```

```
pie(table(bilirubin[,2]))
```

```
pie(table(bilirubin[,3]))
```

```
barplot(table(bilirubin[,1]))
```

```
barplot(table(bilirubin[,2]))
```

```
barplot(table(bilirubin[,3]))
```

„Häufigkeitstabellen“ & Histogramme für quantitative Daten

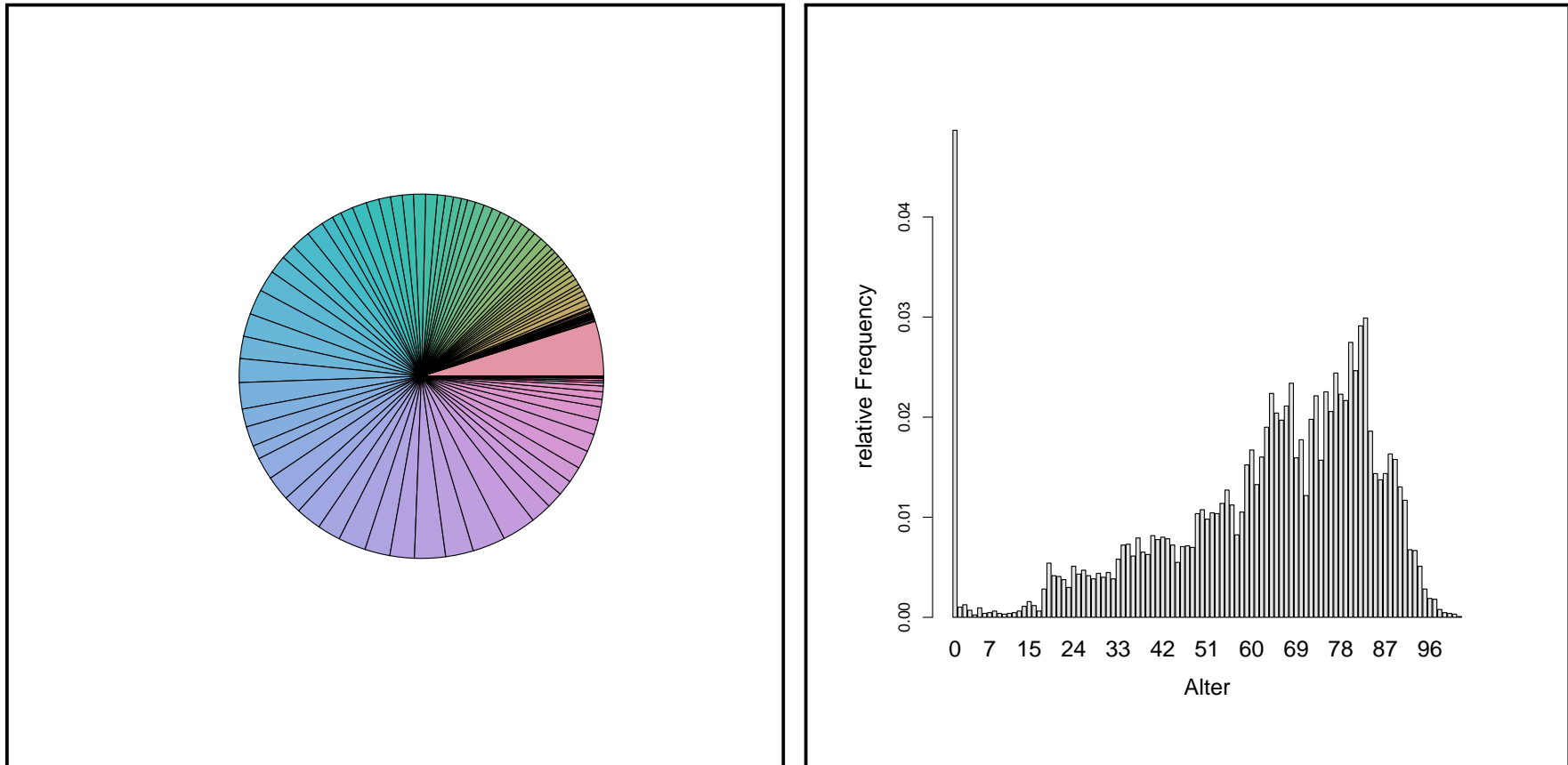
- ▶ Häufigkeitstabelle, Kreis- und Balkendiagramm sind geeignet für **qualitative** Merkmale (nominal und ordinal).
- ▶ Für **quantitativ-diskrete** Merkmale, wenn nicht viele Ausprägungen vorliegen, kann man sie benutzen.
- ▶ Für **quantitative** Merkmale mit sehr vielen Ausprägungen (diskret oder stetig) sind sie unübersichtlich.

Negativbeispiel: Häufigkeiten von *Alter* in den *bilirubin*-Daten:

ALTER	abs. Häufigkeit	rel. Häufigkeit
0	174	0.014
.	.	.
39	90	0.007
40	114	0.008
.	.	.
102	2	.

„Häufigkeitstabellen“ & Histogramme für quantitative Daten

Kreis- und Stabdiagramm sind **nicht** geeignet für **quantitative** Merkmale, wenn sehr viele Ausprägungen vorliegen.

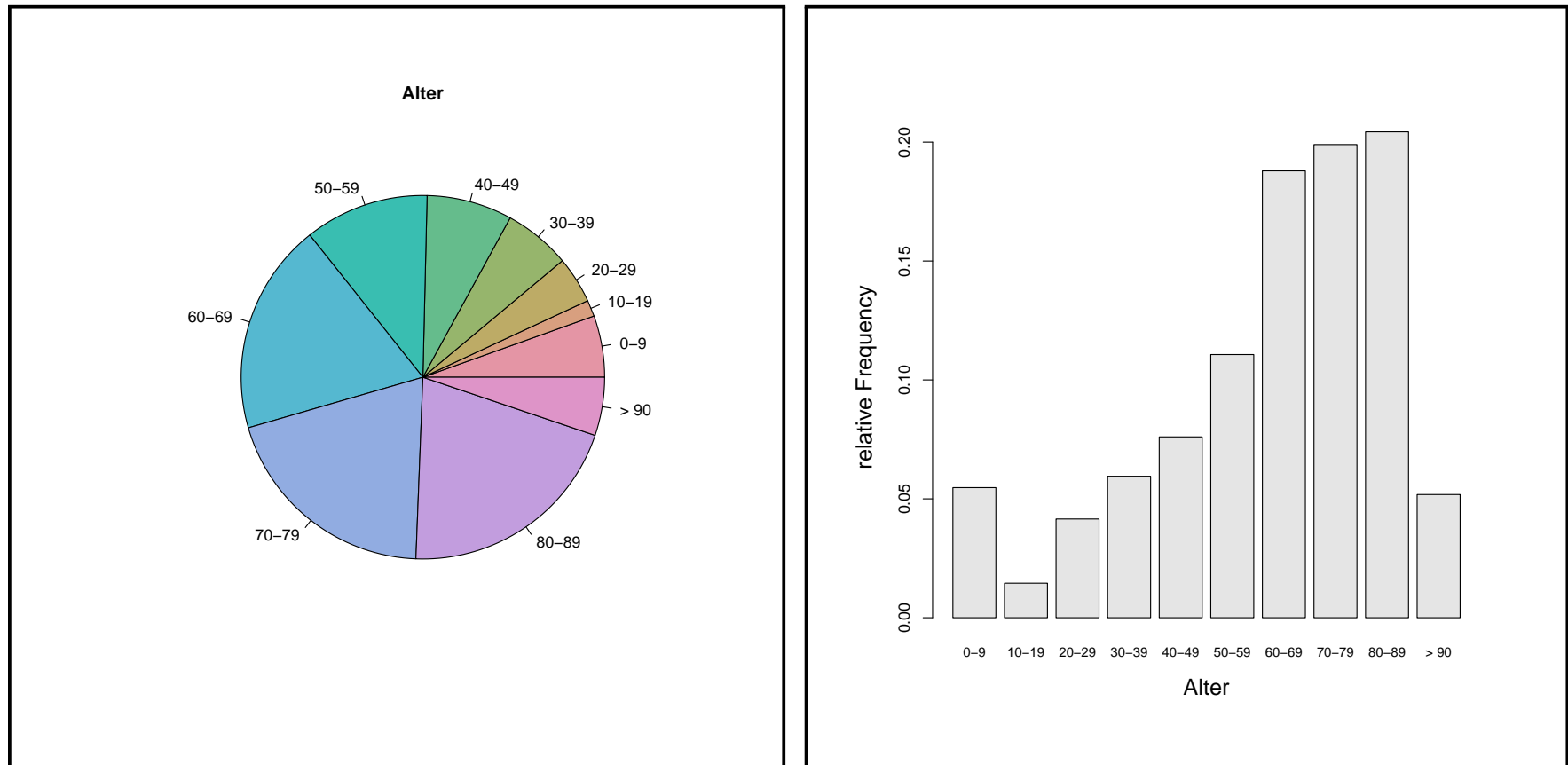


Kreis- und Stabdiagramm der Altersverteilung in den Bilirubin Daten

Häufigkeitstabelle für klassifizierte Merkmale

Durch die Klasseneinteilung der Ausprägungen eines quantitativen Merkmals erzeugt man ein neues **qualitatives** Merkmal.

Alter	Absolute Häufigkeit	Kumulierte absolute Häufigkeit	Relative Häufigkeit	Kumulierte relative Häufigkeit
0-9	697	697	0.0547	0.0547
10-19	185	882	0.0145	0.0692
20-29	529	1411	0.0415	0.111
.
.
80-89	2603	12080	0.204	0.948
> 90	660	12740	0.052	1



Altersverteilung bei den Bilirubin Daten

Beispiel: Klasseneinteilung & grafische Darstellung

Übung 3.16

Bauen Sie die Häufigkeitstabelle, Kreis- und Balkendiagramm aus der vorigen Folie mithilfe der Funktion `cut()` in  nach.

Beispiel: Klasseneinteilung & grafische Darstellung

Übung 3.16

Bauen Sie die Häufigkeitstabelle, Kreis- und Balkendiagramm aus der vorigen Folie mithilfe der Funktion `cut()` in  nach.

```
breakpoints <- c(0, 9, 19, 29, 39, 49, 59, 69, 79, 89, Inf)
bilirubin$ALTERSGRUPPE <- cut(bilirubin[,1],
                              breaks=breakpoints,
                              include.lowest = TRUE,
                              ordered_result = FALSE)
levels(bilirubin$ALTERSGRUPPE) <- c(paste0((0:8)*10, "-"),
                                    (0:8)*10+9, ">90")

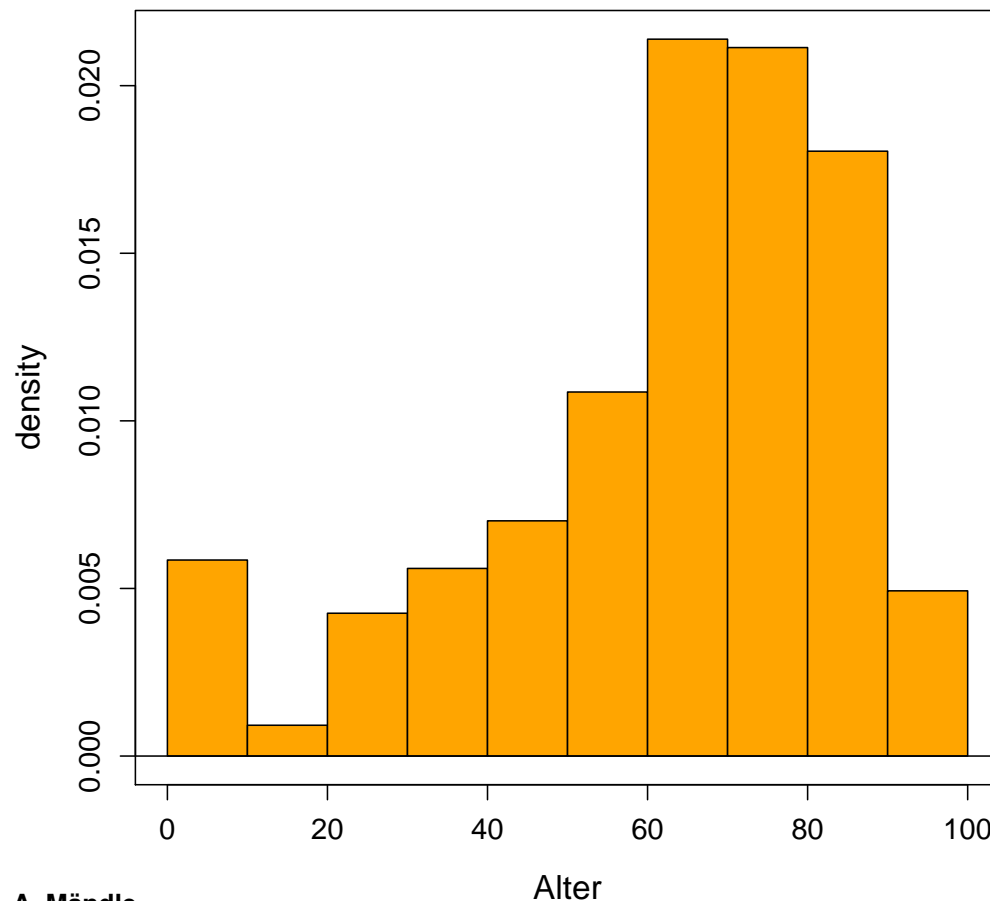
table(bilirubin[,4])

library(RColorBrewer)
pie(table(bilirubin[,4]), main="Altersverteilung",
      col=brewer.pal(n = 10, name = "Spectral"))

barplot(table(bilirubin[,4])/sum(table(bilirubin[,4])),
        xlab="Altersgruppe", ylab="relative_Häufigkeit",
        cex.axis=.6, cex.names=.6)
```


Histogramm: Grafische Darstellung stetiger Merkmale

- ▶ Balkendiagramm: Darstellung der absoluten/relativen Häufigkeiten durch die Balkenhöhe
- ▶ Histogramm: Darstellung der Häufigkeiten durch Flächen (Flächendiagramm)



Beispiel: Histogramm mit hist()

```
hist(bilirubin$ALTER)

# eigene Klassenbreite definieren (breaks) :
summary(bilirubin$ALTER)
hist(bilirubin$ALTER, breaks=seq(0,110,10),
      right=F, col="darkgray")

hist(bilirubin$ALTER, breaks=seq(0,110,10),
      right=F, plot=F)

y<-hist(bilirubin$ALTER, breaks=seq(0,110,10),
        right=F, plot=F)
?hist
y$counts # Anzahl Beobachtungen in Zelle
y$mids   # Mittelpunkte der Zellen
y$breaks # Grenzen der Zellen
y$density # relative Häufigkeiten/Dichteschätzer
```

hist(x,breaks,freq=F)

?hist

x: ein Vektor

breaks: „breakpoints“ zwischen den Histogramm-Zellen

freq: TRUE=absolute Häufigkeiten oder FALSE=Dichteschätzung

right: TRUE=Intervalle rechts abgeschlossen, FALSE=links abgeschlossen

col, main, xlab, xlim,...: Farbe, Titel, x-Achsenbeschriftung, x-Achsen-Wertebereich...

plot: TRUE=Plot erstellen oder FALSE: Daten zurückgeben, siehe unter „Values“:

counts, density, breaks, mids

Echte Histogramme werden nur mit **freq=F** (nicht *default!*) erzeugt.

Histogramme: Übung

Übung 3.17

Arbeiten Sie weiter mit den *bilirubin*-Daten:

1. Erstellen Sie Histogramme für die Altersverteilung jeweils der männlichen und der weiblichen Patienten.
2. Erstellen Sie Histogramme für den Bilirubin-Wert jeweils der männlichen und der weiblichen Patienten.
3. Erstellen Sie Histogramme für den Bilirubin-Wert von jeweils Männern bis einschließlich 18 J. und Männern über 18 Jahre.

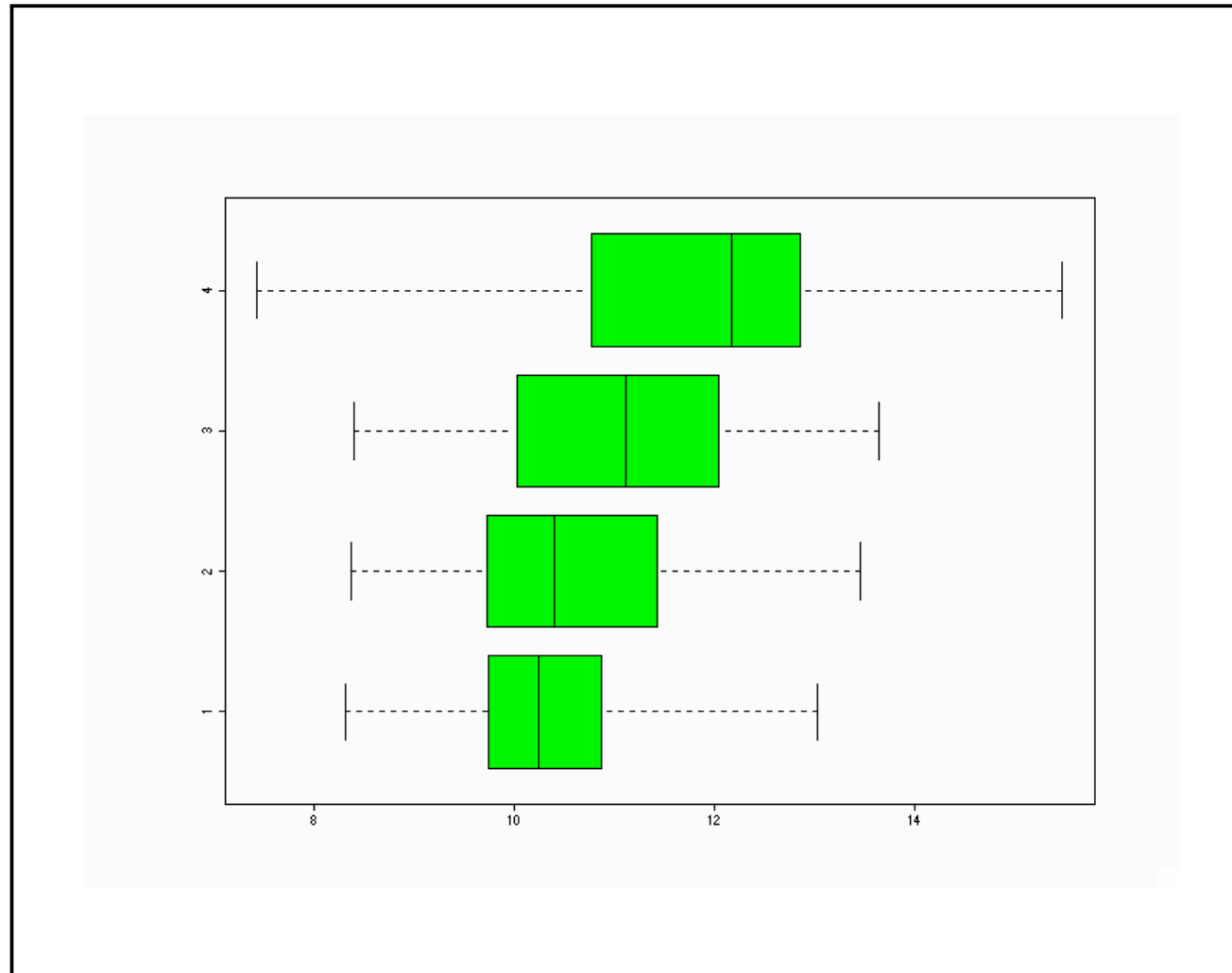
Verwenden Sie: `op <- par(mfrow = c(1, 2))` und `hist()` mit der Option `freq=FALSE`.

Übung 3.17: Lösung

```
bil.m <- bilirubin[bilirubin[,2]=="M",]  
bil.w <- bilirubin[bilirubin[,2]=="W",]  
  
op <- par(mfrow = 1:2)  
hist(bil.w$ALTER, breaks=seq(0,110,10),  
      right=FALSE, col="darkgray", main="Frauen", freq=FALSE)  
hist(bil.m$ALTER, breaks=seq(0,110,10),  
      right=FALSE, col="darkgray", main="Männer", freq=FALSE)  
  
breaks1 <- seq(min(bilirubin$Wert), max(bilirubin$Wert), 0.05)  
hist(bil.w$Wert, breaks=breaks1,  
      right=FALSE, col="darkgray", main="Frauen", freq=FALSE)  
hist(bil.m$Wert, breaks=breaks1,  
      right=FALSE, col="darkgray", main="Männer", freq=FALSE)  
  
bil.m1<-bil.m[bil.m[,1] <= 18 ,]  
bil.m2<-bil.m[bil.m[,1] > 18 ,]  
hist(bil.m1$Wert, breaks=breaks1,  
      right=F, col="darkgray", main="Männer_<=_18", freq=F)  
hist(bil.m2$Wert, breaks=breaks1,  
      right=F, col="darkgray", main="Männer_>_18", freq=F)  
par(op)
```

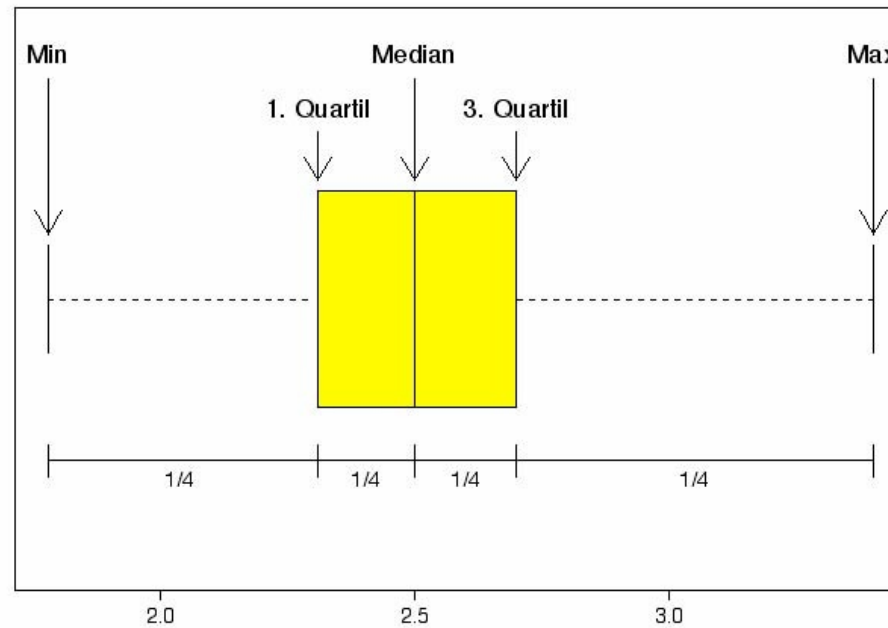
Boxplots

Graphische Darstellung der Verteilungen von metrisch-skalierten Merkmalen durch Boxplots:



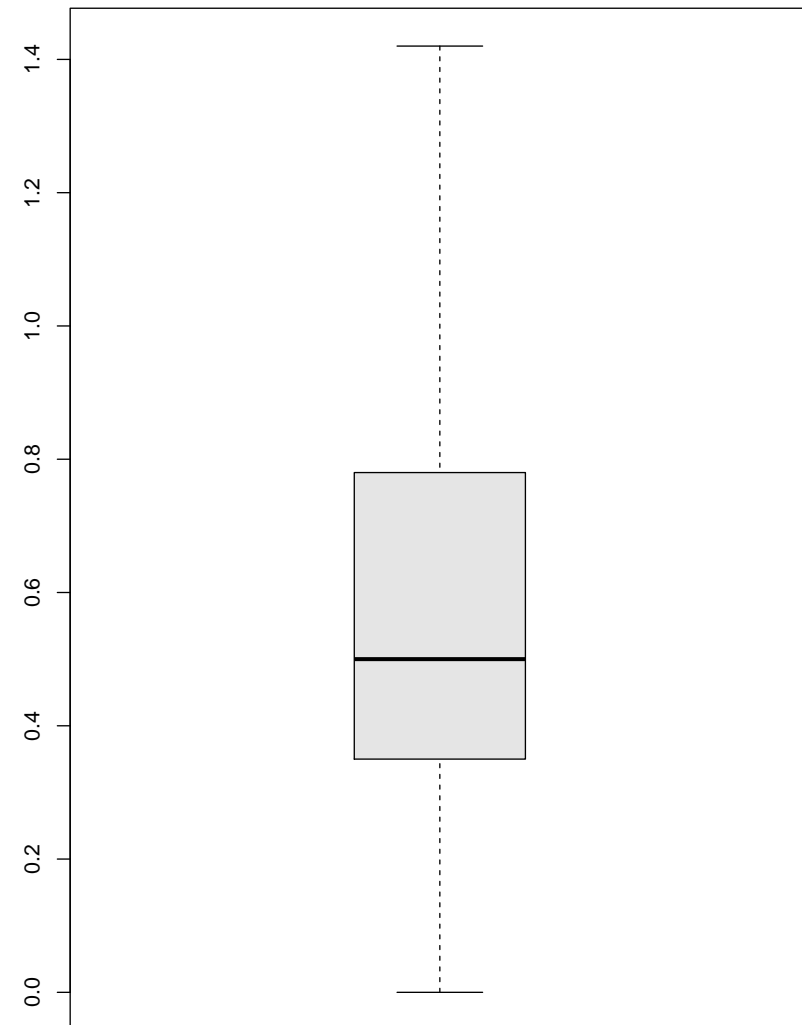
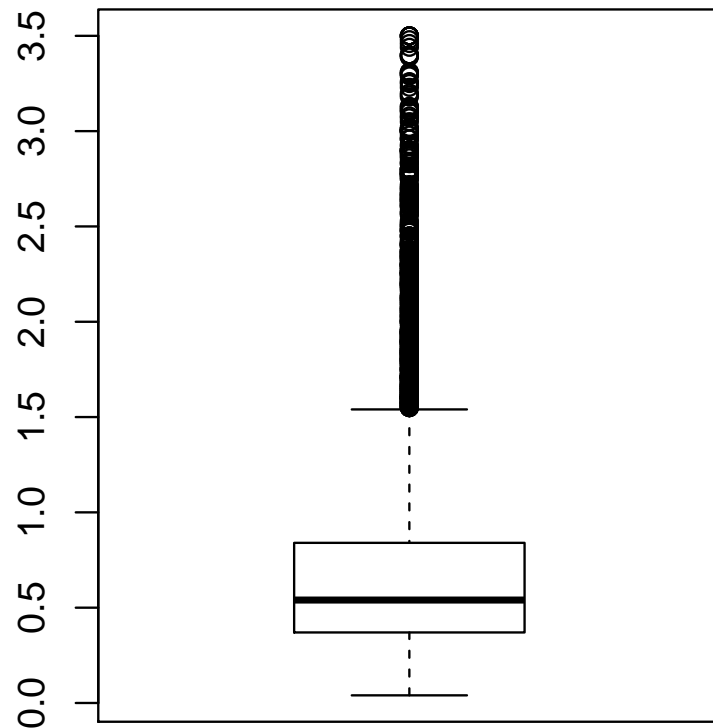
Boxplot: Beispiel

$$x_{min} = 1.7, x_{max} = 3.4, x_{med} = 2.5, x_{0.25} = 2.3, x_{0.75} = 2.7$$



Boxplots mit *boxplot()*

```
boxplot(bilirubin[,3])  
boxplot(x=bilirubin[,3], outline=FALSE, col="gray90", boxwex=0.5)
```



boxplot(...,outline=T,range=1.5)

?boxplot

range: bestimmt in welchem Bereich die Whiskers liegen – Whiskers erstrecken sich zu den äußersten Datenpunkten, die weniger als $\text{range} \times \text{IQR}$ von der Box entfernt sind. Ausnahme, bei $\text{range}=0$ liegen die Whiskers auf den extremen Datenpunkten.

outline: FALSE bedeutet Ausreißer werden nicht gezeichnet.

```
boxplot(x = bilirubin[,3], outline=TRUE,
        col="gray90", boxwex=0.5,
        range=1.5) # entspricht default range
boxplot(x=bilirubin[,3], outline=FALSE,
        col="gray90", boxwex=0.5)
boxplot(x=bilirubin[,3], outline=FALSE,
        col="gray90", boxwex=0.5, range=0)
```

Boxplots mit *ggplot2*

```
geom_boxplot(outlier.colour="black", outlier.shape=16,  
             outlier.size=2, notch=FALSE)
```

notch: TRUE: gekerbter Boxplot, stellt auch Konfidenzintervalle für den Median dar.

outlier.colour, outlier.shape, outlier.size: Farbe, Form und Größe der Punkte.

```
library(ggplot2)  
p <- ggplot(bilirubin, aes(x="", y=Wert)) +  
  geom_boxplot(); p  
p + geom_boxplot(outlier.colour="red", outlier.shape=8,  
                outlier.size=4)  
  
p <- ggplot(bilirubin, aes(x=SEX, y=Wert, color=SEX)) +  
  geom_boxplot(); p  
p + scale_color_brewer(palette="Dark2")  
p <- ggplot(bilirubin, aes(x=SEX, y=Wert, fill=SEX)) +  
  geom_boxplot(); p  
p + scale_fill_brewer(palette="Accent") + coord_flip()
```

Boxplot: Präsenzaufgabe 2

Übung 3.18

Laden Sie den Datensatz *iris* und betrachten Sie die ersten 10 Zeilen dieser Datei.

- a. Erstellen Sie Histogramme für jeweils die 4 metrisch-skalierten Merkmale des Datensatzes.
- b. Erstellen Sie Boxplots jeweils für die 4 metrisch-skalierten Merkmale des Datensatzes.
- c. Erstellen Sie gruppierte Boxplots jeweils für die 4 metrisch-skalierten Merkmale gruppiert nach dem Merkmal *species*.

Charakteristisches Maßzahlen für quantitative Daten

Lage- und Streuungsmaß für metrisch-skalierte Merkmale:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \left(\frac{\text{Summe}}{\text{Anzahl}} \right)$$

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

IQR= Interquantilabstand = (3. Quartil) - (1. Quartil)

R= Spannweite = (größter Wert) - (kleinster Wert) = $x_{max} - x_{min}$

Charakteristisches Maßzahlen: Übung

Übung 3.19

- 1) Definieren Sie einen Vektor x und berechnen Sie mit den Funktionen $mean(x)$, $sd(x)$, $var(x)$, $IQR(x)$ und $range(x)$ die entsprechenden Maßzahlen.
- 2) Berechnen Sie die o.g. Maßzahlen für die Variablen *Alter* und *Wert* aus den *bilirubin*-Daten.
- 3) Der Variationskoeffizient (cv) wird berechnet als Standardabweichung dividiert durch den arithmetischen Mittelwert und ist ein *relatives Streuungsmaß*. Schreiben Sie eine Funktion, die den cv berechnet.

summary()

Bisher haben wir für einzelne Lagemaße einzelne Funktionen durchgeführt. Eine Auswahl von wichtigen Lagemaßen kann man mit der Funktion *summary()* ausgeben:

```
summary(bilirubin[,1])
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.00	52.00	68.00	63.77	80.00	102.00

```
summary(bilirubin[,3])
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.00	0.35	0.50	0.653	0.78	3.5

3. Deskriptive Statistik für:

1. ein qualitatives Merkmal (univariat)
2. zwei qualitative Merkmale (bivariat)
3. ein quantitatives Merkmal (univariat)
4. **ein qualitatives und ein quantitatives Merkmal (bivariat)**
5. zwei quantitative Merkmale (bivariat)

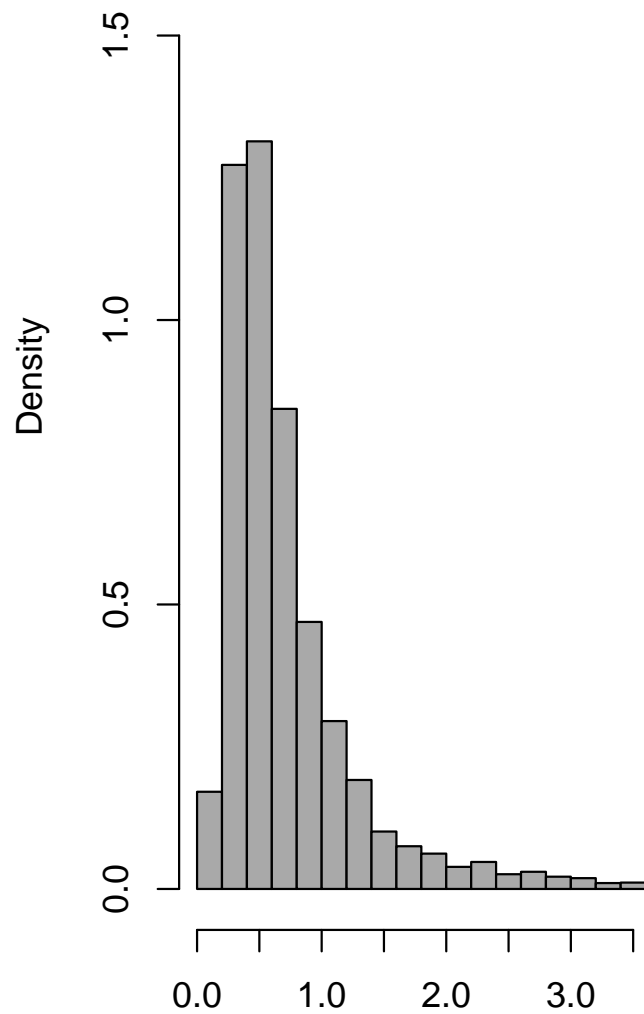
Gruppiertes Histogramm

```
bilirubin <- read.table(file="Daten/bilirubin.txt",
                        head=TRUE, sep=";")
bil.m <- bilirubin[bilirubin[,2]=="M",]
bil.w <- bilirubin[bilirubin[,2]=="W",]

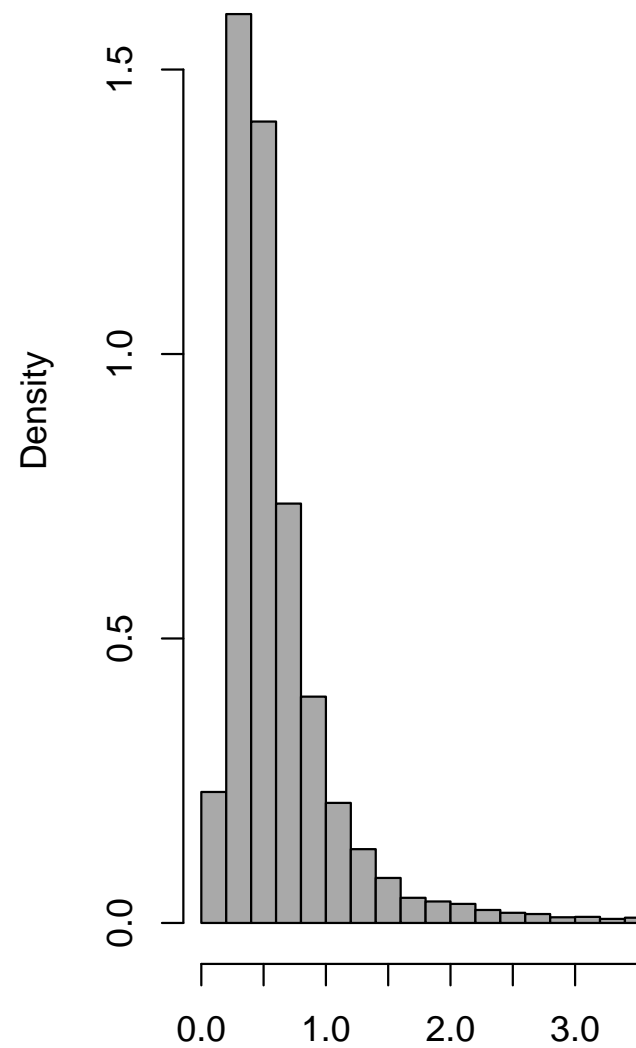
op <- par(mfrow = c(1,2))
hist(bil.m$Wert, right=FALSE,
     col="darkgray", freq=FALSE,
     ylim=c(0,1.8))
hist(bil.w$Wert, right=FALSE,
     col="darkgray", freq=FALSE,
     ylim=c(0,1.8))
par(op)
```


Gruppiertes Histogramm

Histogram of bil.m\$Wert



Histogram of bil.w\$Wert

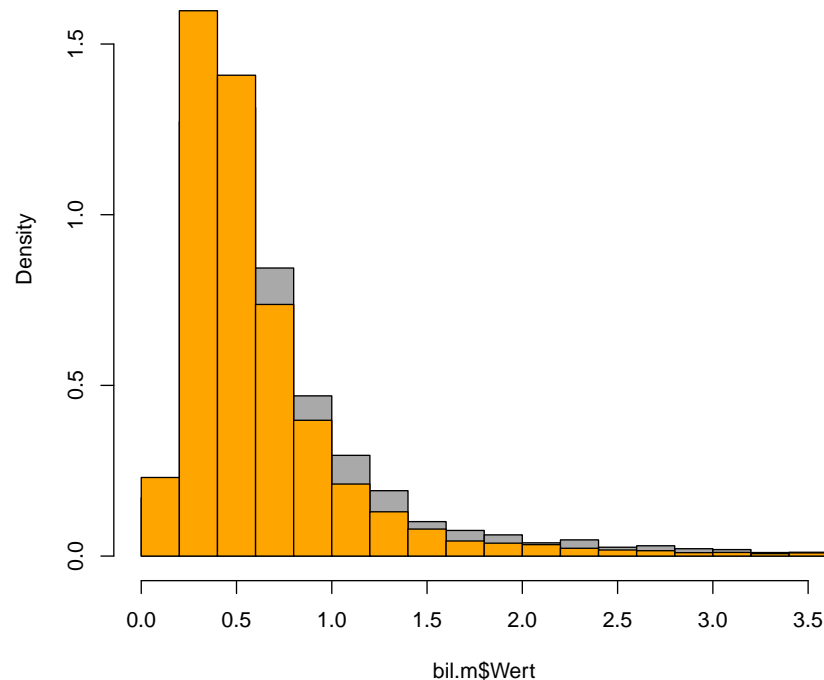


Gruppiertes Histogramm

Per Argument **add=TRUE** zeichnet man in einen bestehenden Plot:

```
hist(bil.m$Wert, right=FALSE, col="darkgray",  
      freq=FALSE, ylim=c(0,1.8))  
hist(bil.w$Wert, right=FALSE, col="orange",  
      freq=FALSE, add=TRUE)
```

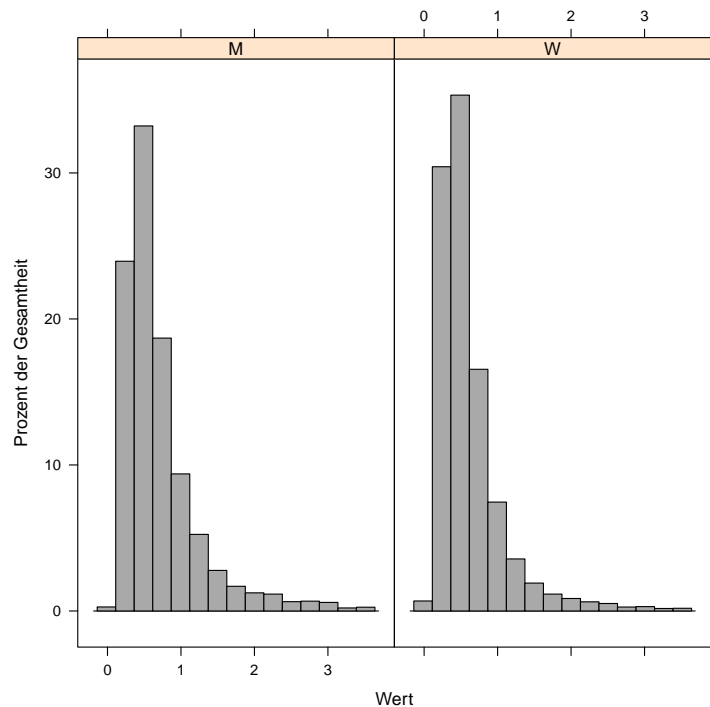
Histogram of bil.m\$Wert



Gruppiertes Histogramm: `lattice::histogram()`

Die Funktion `histogram()` aus dem Package *lattice* zeichnet gruppierte Histogramme:

```
install.packages("lattice")  
library(lattice)  
histogram(~ Wert | SEX, data = bilirubin,  
          col="darkgray", type = "density")
```



Histogramme mit *ggplot2*

```
geom_histogram()
```

data: Fall andere als die geladenen Daten verwendet werden sollen.

binwidth, bins, breaks: Breite der Bins, deren Anzahl, alternativ Breakpoints angeben.

```
# Einfaches Histogramm
```

```
ggplot(bil.m, aes(x=Wert)) + geom_histogram()
```

```
# Andere Bin-Breite
```

```
p <- ggplot(bil.m, aes(x=Wert)) +  
  geom_histogram(binwidth=0.2); p
```

```
# andere Farben - Histogramm über das Alte geplottet (!)
```

```
p + geom_histogram(color="black", fill="white")
```

```
# Füllfarben je nach Gruppe
```

```
ggplot(bilirubin, aes(x=Wert, fill=SEX, color=SEX)) +  
  geom_histogram(position="identity")
```

```
# Halbtransparente Füllfarbe
```

```
p <- ggplot(bilirubin, aes(x=Wert, fill=SEX, color=SEX)) +  
  geom_histogram(position="identity", alpha=0.5)
```

```
p
```

A. Mändle

Histogramm und Dichtefunktion

Histogramme eignen sich gut für die Darstellung der Verteilung eines einzelnen Merkmals.

Wenn man zwei Verteilungen vergleichen will, sind sie ungeeignet, da sich die Balken überdecken.

Dann verwendet man z.B. **Dichtepolygone** oder *geschätzte Dichtefunktionen*.

Die universelle Plotfunktion *plot()*

```
plot(x,y, type="p")
```

type="p": Art des Plots, z.B.:

- *"p" Punkte,*
- *"l" Linien,*
- *"b" Punkte und Linien,*
- *...*
- *"n" kein Plotten.*

Beispiele:

```
plot(1:10, 21:30, "l")
```

```
x <- seq(0, 2*pi, pi/10)
```

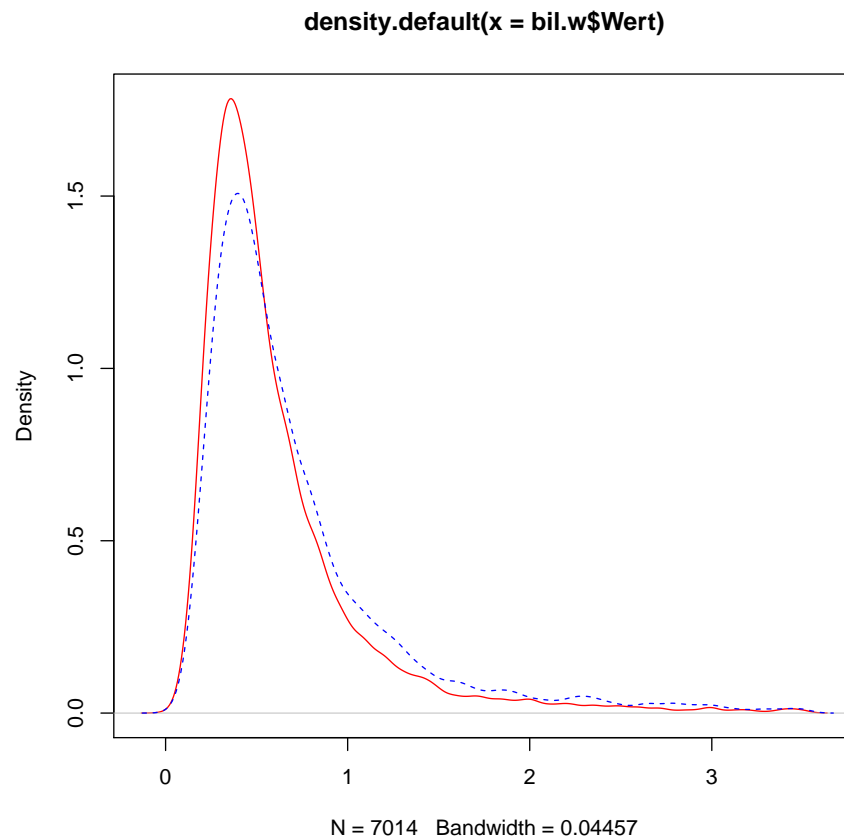
```
y <- sin(x)
```

```
plot(x, sin(x), "b")
```

(stetige) Dichteschätzer: *density()*

Einen geglätteten Dichteschätzer erhält man durch die **density()**-Funktion, die man mittels **plot()** visualisieren kann:

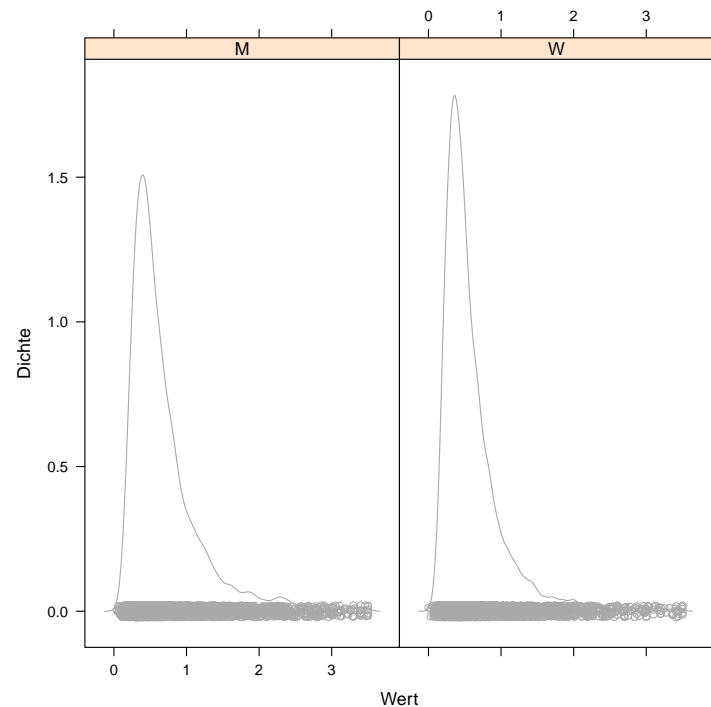
```
plot(density(x=bil.w$Wert), col="red", lty=1)  
lines(density(x=bil.m$Wert), col="blue", lty=2)
```



Gruppierte Dichteschätzer

Die Funktion *densityplot* aus dem Package *lattice* zeichnet gruppierte Dichteschätzer:

```
install.packages("lattice")  
library(lattice)  
densityplot(~ Wert | SEX, data = bilirubin, col="darkgray")
```



(Gruppierte) Dichteschätzer mit ggplot2

Die Funktion `geom_density` zeichnet Dichteschätzer in `ggplot2`:

```
# Einfacher Dichteschätzer
```

```
ggplot(bil.w) + geom_density(aes(x = Wert))
```

```
# Farbe Rot
```

```
ggplot(bil.w) + geom_density(aes(x = Wert), color="red")
```

```
# Gruppierte Dichteschätzer
```

```
ggplot(bilirubin) +
```

```
  geom_density(aes(x = Wert, color = SEX))
```

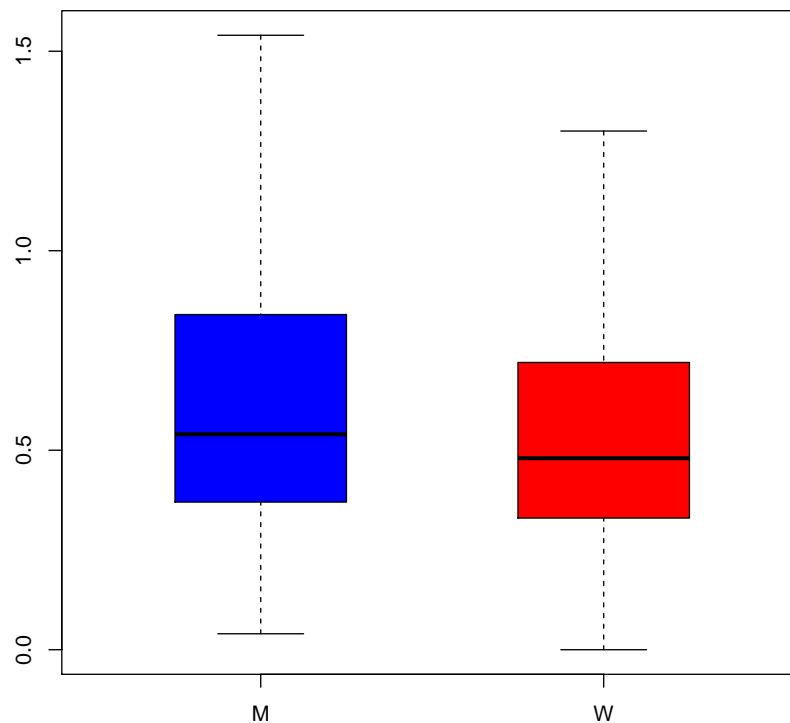
```
# gefüllte, transparente Flächen
```

```
ggplot(bilirubin) +
```

```
  geom_density(aes(x = Wert, fill = SEX),  
               alpha = 0.2)
```

Gruppierte Boxplots

```
boxplot(bilirubin[,3] ~ bilirubin[,2], data=bilirubin)
boxplot(bilirubin[,3] ~ bilirubin[,2], data=bilirubin,
        col=c("blue", "red"), boxwex=0.5, outline=FALSE,
        ylab="Bilirubin_Values")
```



Übung 3.20

Erstellen Sie 4 Boxplots für 4 metrisch-skalierte Merkmale gruppiert nach dem Merkmal „Species“ beim Datensatz *iris*.

Übung 3.20

Erstellen Sie 4 Boxplots für 4 metrisch-skalierte Merkmale gruppiert nach dem Merkmal „Species“ beim Datensatz *iris*.

```
iris[1:10,]
```

```
for(i in 1:4){  
  hist(iris[,i],main=colnames(iris[i]))  
}
```

```
for(i in 1:4){  
  boxplot(iris[,i]~iris[,5],main=colnames(iris[i]))  
}
```

Beispiel zu: density()

Dichte der metrisch-skalierten Merkmalen aus dem **iris**-Datensatz gruppiert nach dem Merkmal „Species“ darstellen:

```
iris1<-split(iris,iris[,5])
seto <-iris1[[1]]
vers <-iris1[[2]]
virg <-iris1[[3]]
for(i in 1:4){
  plot(density(seto[,i]), main=colnames(seto[i]), col="blue")
  lines(density(vers[,i]), col="red")
  lines(density(virg[,i]), col="green")
}
# Problem mit Range der x-Werte!
```

Beispiel zu: density()

```
for(i in 1:4){  
  plot(density(seto[,i]), main=colnames(seto[i]), col="blue",  
        xlim=range(seto[,i], vers[,i], virg[,i]))  
  lines(density(vers[,i]), col="red")  
  lines(density(virg[,i]), col="green")  
}  
# Problem mit Wertebereich der y-Achse!
```

Beispiel zu: density()

```
for(i in 1:4){
  y1<-density(seto[,i])$y
  y2<-density(vers[,i])$y
  y3<-density(virg[,i])$y
  Y<-max(y1,y2,y3)
  #windows()
  plot(density(seto[,i]),main=colnames(seto[i]),col="blue",
        xlim=range(seto[,i],vers[,i],virg[,i]),ylim=c(0,Y))
  lines(density(vers[,i]),col="red")
  lines(density(virg[,i]),col="green")
}
```

Übung 3.21

Berechnen Sie *mean()*, *sd()*, *var()*, *IQR()*, *range()* für die Variablen *Alter* und *Wert* aus den *bilirubin*-Daten für Männer und Frauen.

3. Deskriptive Statistik für:

1. ein qualitatives Merkmal (univariat)
2. zwei qualitative Merkmale (bivariat)
3. ein quantitatives Merkmal (univariat)
4. ein qualitatives und ein quantitatives Merkmal (bivariat)
5. **zwei quantitative Merkmale (bivariat)**

I Graphische Darstellung: Streudiagramm (scatterplot) ist ein Hilfsmittel, um den Zusammenhang zwischen 2 metrischen Merkmalen zu veranschaulichen.

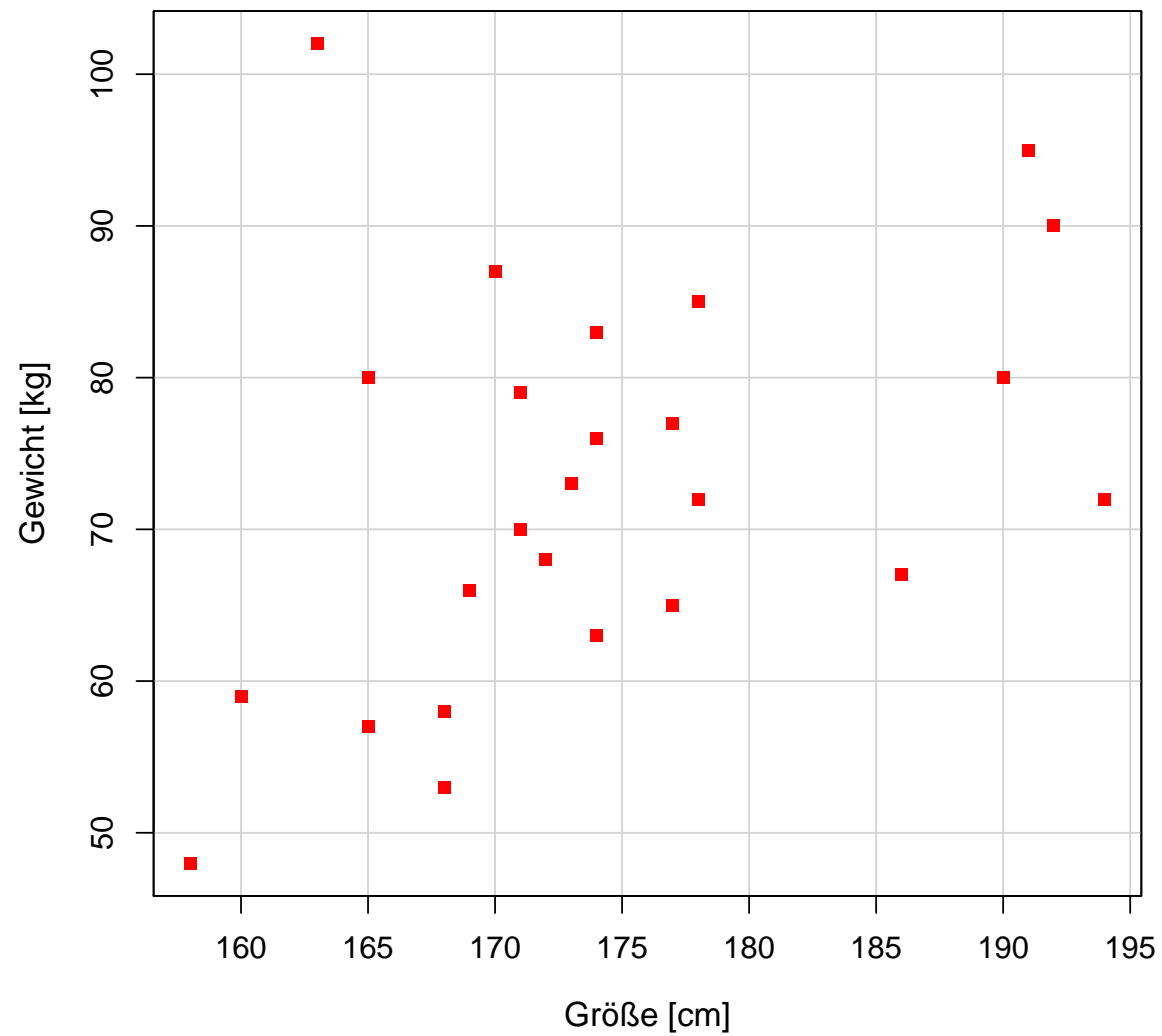
II Charakteristische Maßzahlen: Korrelationskoeffizienten quantifizieren die Stärke und das Ausmaß des Zusammenhangs.

Streudiagramm/Scatterplot

Jedes Paar (X_i, Y_i) wird durch einen Punkt in der XY – Ebene repräsentiert. Beispiel: Körpergröße und Gewicht

i	Person	x_i (cm)	y_i (kg)	i	Person	x_i (cm)	y_i (kg)
1	Magda	158	48	14	Jörg	173	73
2	Anna	160	59	15	Volker	174	76
3	Roland	163	102	16	Stefan	174	63
4	Swetlana	165	57	17	Heike	174	83
5	Alexander	165	80	18	Karpo	177	65
6	Tamara	168	53	19	Vladimir	177	77
7	Iwan	168	58	20	Stanislaw	178	72
8	Eva	168	58	21	Felix	178	85
9	Karoline	169	66	22	Andrej	186	67
10	Nikolaj	170	87	23	Walerij	190	80
11	Alexandra	171	70	24	Jost	191	95
12	Ingrid	171	79	25	Stefan	192	90
13	Oksana	172	68	26	Witalij	194	72

Streudiagramm: Größe und Gewicht (n=26)



Streudiagramme

Streudiagramme kann man mit der Funktion *plot()* erstellen:

plot(x,y) # plottet 2 Variablen

Wir laden den Datensatz *cars* aus dem Paket *datasets*.

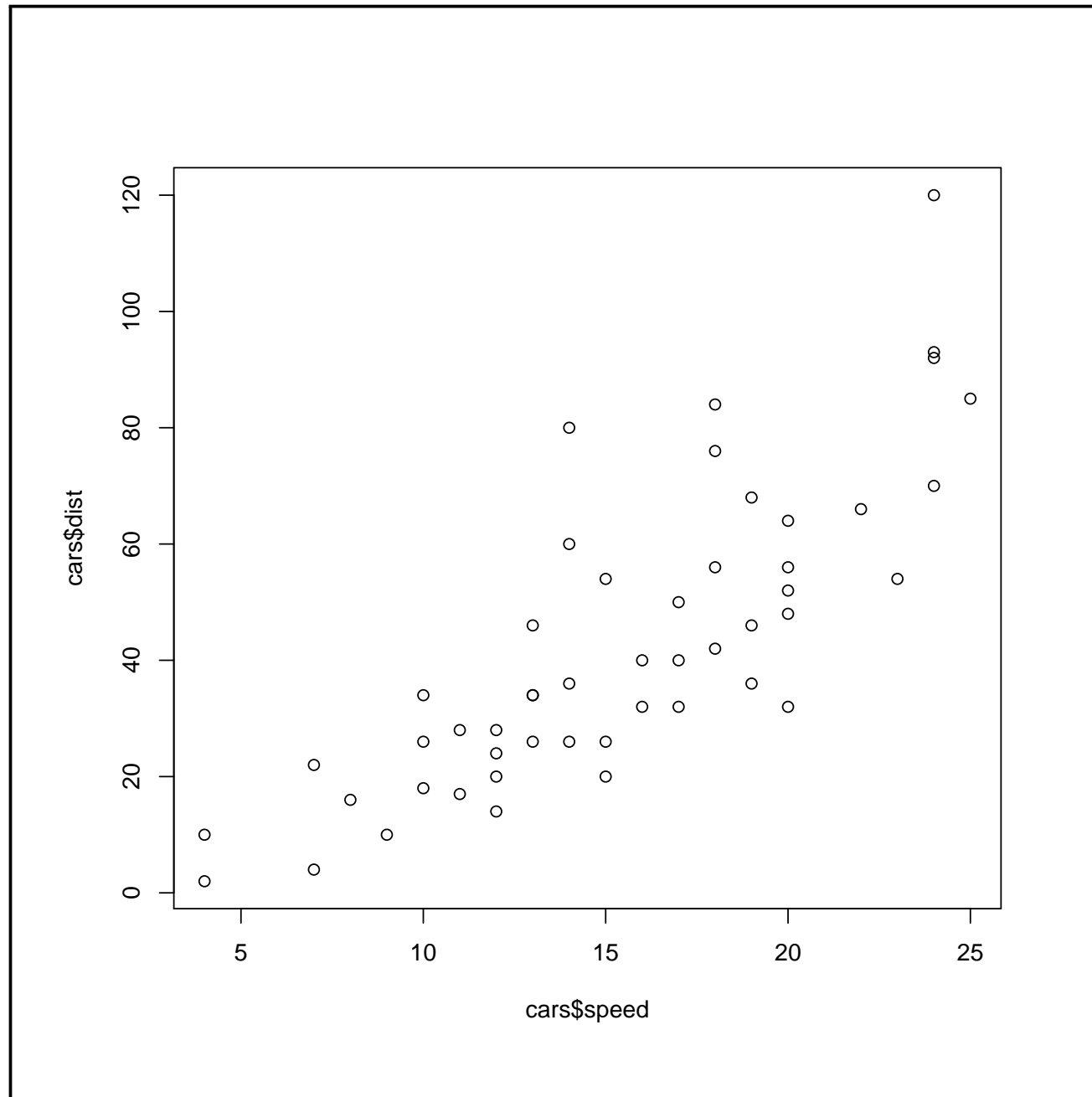
?cars: *The data give the speed of cars and the distances taken to stop. Note that the data were recorded in the 1920s.*

```
cars[1:4, ]
```

	speed	dist
1	4.00	2.00
2	4.00	10.00
3	7.00	4.00
4	7.00	22.00

```
plot(cars$speed, cars$dist)
```

Streudiagramme



Streudiagramme: plot()

`plot(x,y,xlim=range(x),ylim=range(y),type="p",...)`

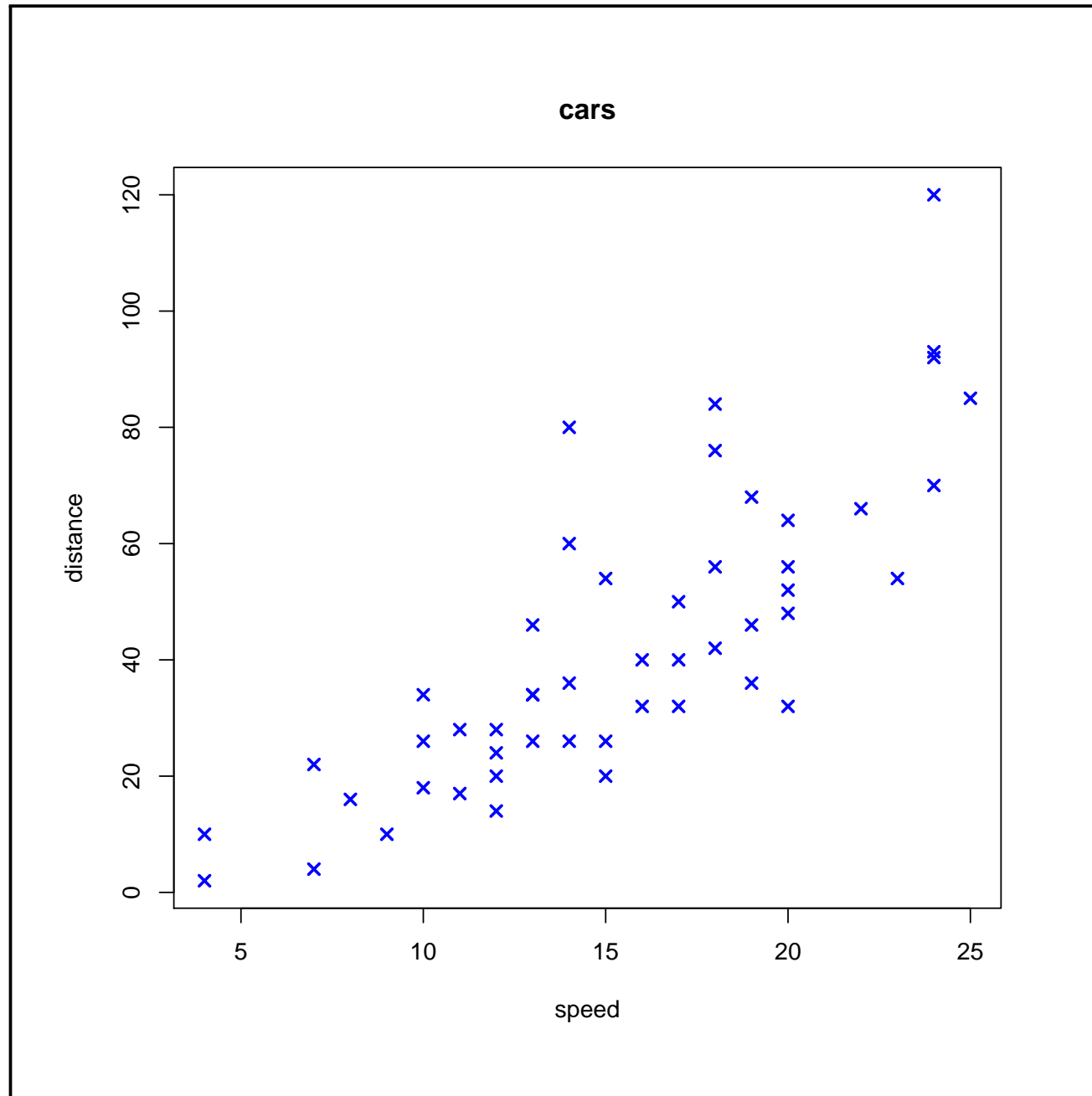
Das Argument x (und eventuell y) ist erforderlich und die restliche optional.

type: Art der Grafik, "p": Punkte, "l": Linie, "b": Punkte + Linie

pch: 0:18, Index für das verwendete Symbol für die Darstellung von Datenpunkten, vgl.: de.wikibooks.org/wiki/GNU_R:_plot

Beispiel:

```
plot(x=cars$speed, y=cars$dist, pch=4, lwd=2, col="blue",  
      xlab="speed", ylab="distance", main="cars")
```



Streudiagramme: Übung

Übung 3.22

Erstellen Sie Streudiagramme für jeweils 2 Merkmale aus dem *iris*-Datensatz und modifizieren Sie dabei die folgenden Optionen: *cex*, *cex.axis*, *cex.main*, *col.axis*, *pch*.

Streudiagramme: Übung

Übung 3.22

Erstellen Sie Streudiagramme für jeweils 2 Merkmale aus dem *iris*-Datensatz und modifizieren Sie dabei die folgenden Optionen: *cex*, *cex.axis*, *cex.main*, *col.axis*, *pch*.

```
plot(iris[,1], iris[,2], col = "red", pch = 2,  
     main = "Kelchblatt",  
     cex=1, cex.axis=0.8,  
     cex.main=2, col.axis=4,  
     xlab="Länge", ylab="Breite")
```

Gruppiertes Streudiagramm: Übung

Übung 3.23

Erstellen Sie Streudiagramme für jeweils 2 Merkmale aus dem *iris*-Datensatz gruppiert nach dem Merkmal „Species“.

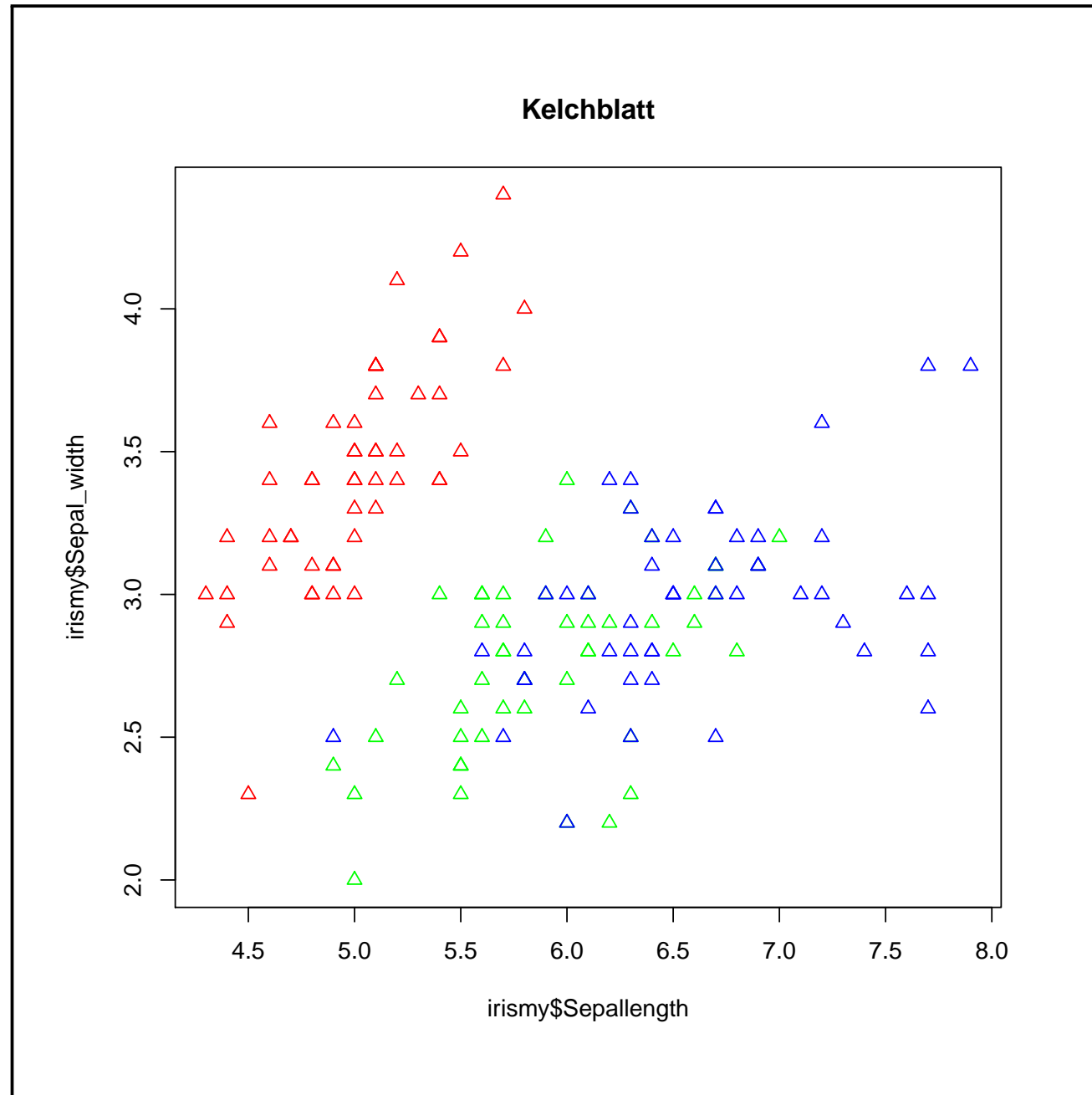
Übung: Lösung

```
plot(iris$Sepal.Length, iris$Sepal.Width, main = "Kelchblatt",  
     col = c("red", "green", "blue")[iris$Species], pch = 2)
```

```
plot(iris$Sepal.Length, iris$Petal.Length,  
     col = c("red", "green", "blue")[iris$Species], pch = 3)
```

```
plot(iris$Petal.Length , iris$Petal.Width,  
     col = c("red", "green", "blue")[iris$Species], pch = 15)
```

```
plot(iris$Sepal.Length, iris$Sepal.Width, main = "Kelchblatt",  
     col = c("red", "green", "blue")[iris$Species],  
     xlab="Sepal_length_[cm]", ylab="Sepal_width_[cm]")
```



Streudiagramme mit *ggplot2*

```
geom_point(size, color, shape)
```

size, color, shape: Größe, Farbe und Form der Punkte

```
# Einfacher Scatter Plot
```

```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width)) + geom_point()
```

```
# Grösse und Form ändern
```

```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width)) +  
  geom_point(size=2, shape=23)
```

```
# Gruppiertes Scatter Plot - Form der Punkte je nach Species
```

```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width,  
                 shape=Species)) + geom_point()
```

```
# Form und Farbe ändern
```

```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width,  
                 shape=Species, color=Species)) + geom_point()
```

```
# Grösse, Farbe und Form ändern
```

```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width,  
                 shape=Species, color=Species, size=Species)) +  
  geom_point()
```

Streudiagramme: Übung

Übung 3.24

Modifizieren Sie den Befehl zur Darstellung des Streudiagramms für den *iris*-Datensatz so, dass sich sowohl Farbe als auch Symbol für die Darstellung der Datenpunkte je nach *species* unterscheiden.

Streudiagramme: Übung

Übung 3.24

Modifizieren Sie den Befehl zur Darstellung des Streudiagramms für den *iris*-Datensatz so, dass sich sowohl Farbe als auch Symbol für die Darstellung der Datenpunkte je nach *species* unterscheiden.

```
plot(iris$Sepal.Length, iris$Sepal.Width,  
col = c("red", "green", "blue")[iris$Species],  
pch = c(2, 3, 15)[iris$Species], main = "Kelchblatt",  
xlab="Sepal_length_[cm]", ylab="Sepal_width_[cm]")
```

Streudiagramm-Matrix

Enthält ein Datensatz mehr als 2 metrisch-skalierte Merkmale, so können ihre paarweisen gemeinsamen (empirischen) Verteilungen in einer Streudiagramm-Matrix veranschaulicht werden. Beispiele:

```
plot(iris[1:4]) # oder  
pairs(iris[1:4])
```

Übung 3.25

Modifizieren Sie den Befehl oben so; dass die Farbe und/oder das Symbol für die Darstellung der Datenpunkte sich je nach *species* unterscheidet.

Streudiagramm-Matrix

Enthält ein Datensatz mehr als 2 metrisch-skalierte Merkmale, so können ihre paarweisen gemeinsamen (empirischen) Verteilungen in einer Streudiagramm-Matrix veranschaulicht werden. Beispiele:

```
plot(iris[1:4]) # oder  
pairs(iris[1:4])
```

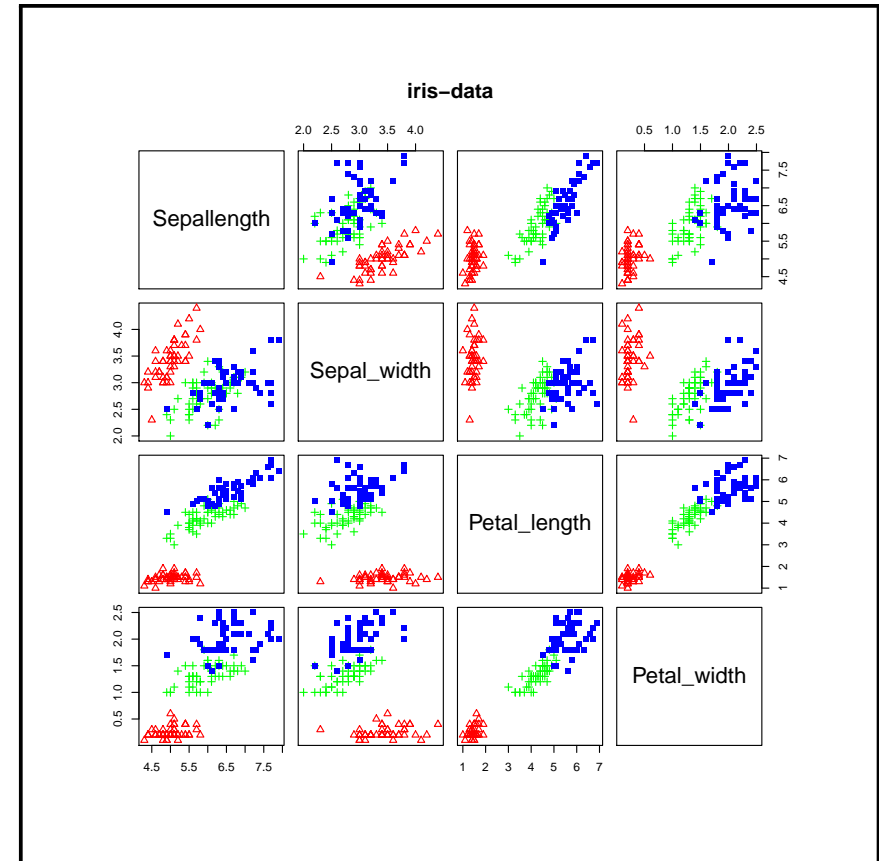
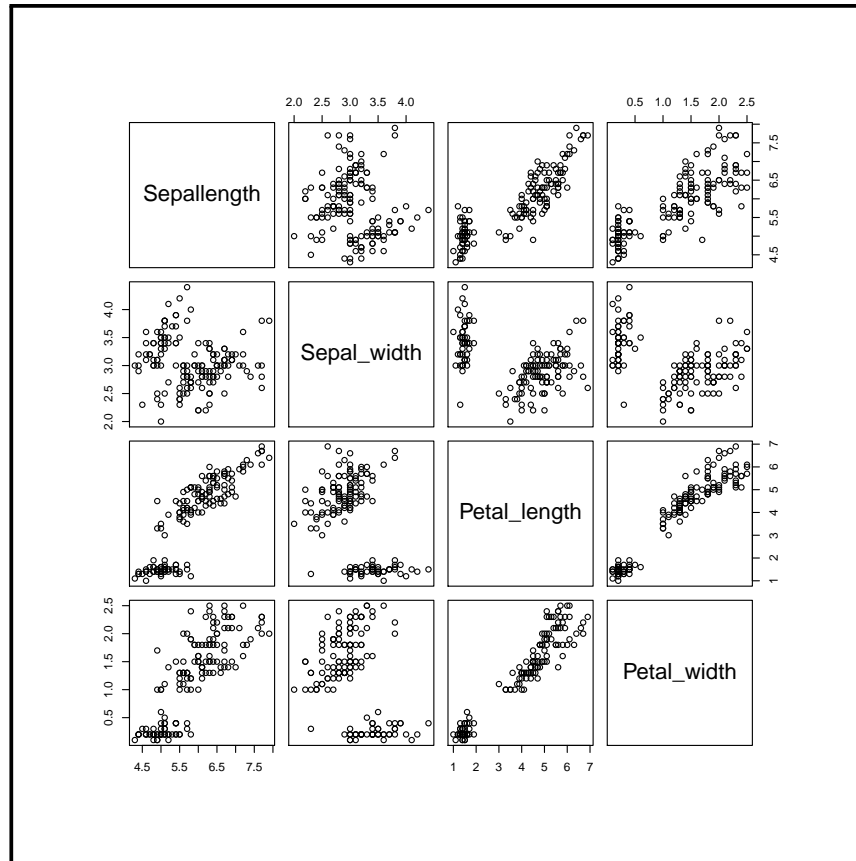
Übung 3.25

Modifizieren Sie den Befehl oben so; dass die Farbe und/oder das Symbol für die Darstellung der Datenpunkte sich je nach *species* unterscheidet.

```
plot(iris[1:4],  
col = c("red", "green", "blue")[iris$Species],  
pch = c(2, 3, 15)[iris$Species], main = "iris-data")
```

Im ggplot-Look:

```
library(GGally)  
ggpairs(iris, aes(colour = Species, alpha = 0.4))
```



Charakteristische Maßzahlen für zwei quantitative Merkmale

(empirische) Kovarianz und Korrelation sind deskriptive Maßzahlen für den Zusammenhang zweier Merkmale.

Definition 3.1

Sei (x, y) eine zweidimensionale Stichprobe vom Umfang n :

$$x = (x_1, x_2, \dots, x_n)$$

$$y = (y_1, y_2, \dots, y_n)$$

dann ist die Stichprobenkovarianz (oder empirische Kovarianz) definiert als:

$$s_{xy} = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

mit

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

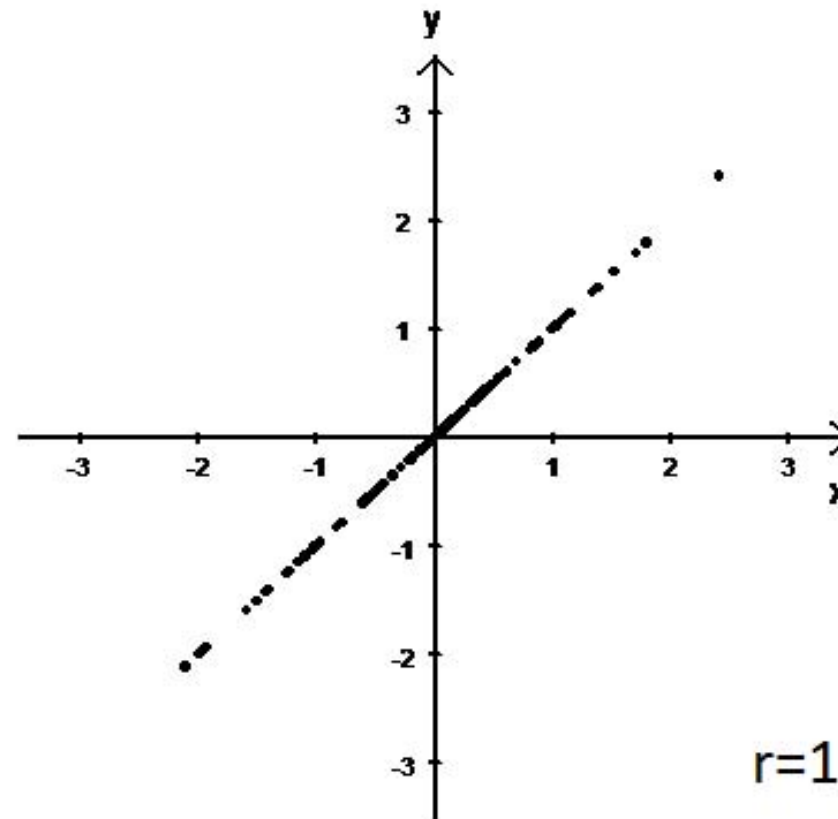
Pearson-Korrelationskoeffizient

Definition 3.2

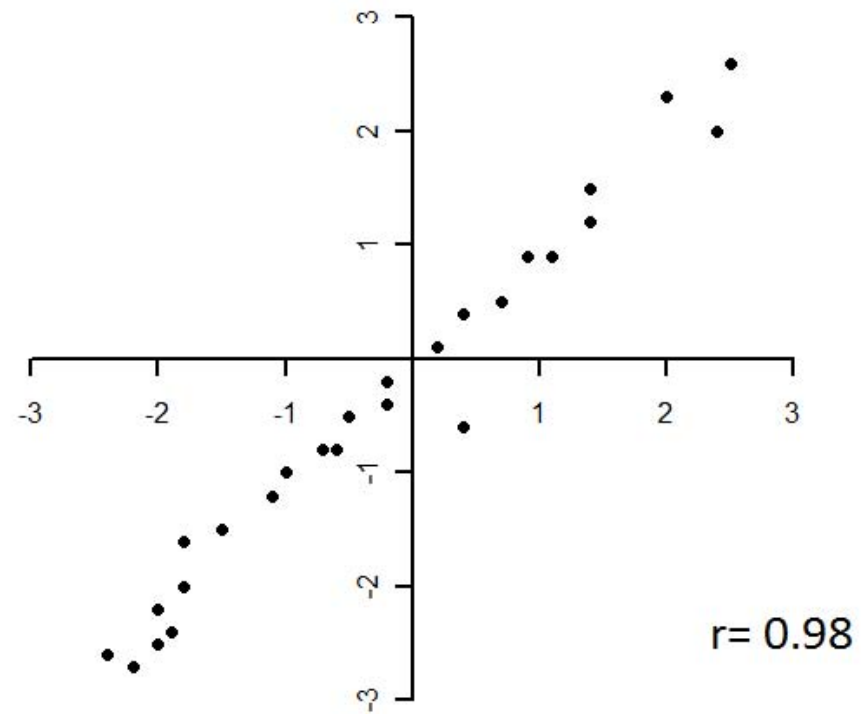
Es Seien (x, y) und s_{xy} wie bei Definition (3.1). Weiter seien $s_x^2 > 0$ und $s_y^2 > 0$ (s_x^2 und s_y^2 : empirische Varianz von X und Y). Dann ist der empirische Pearson-Korrelationskoeffizient definiert als:

$$r_{xy} = \frac{s_{xy}}{s_x s_y}$$

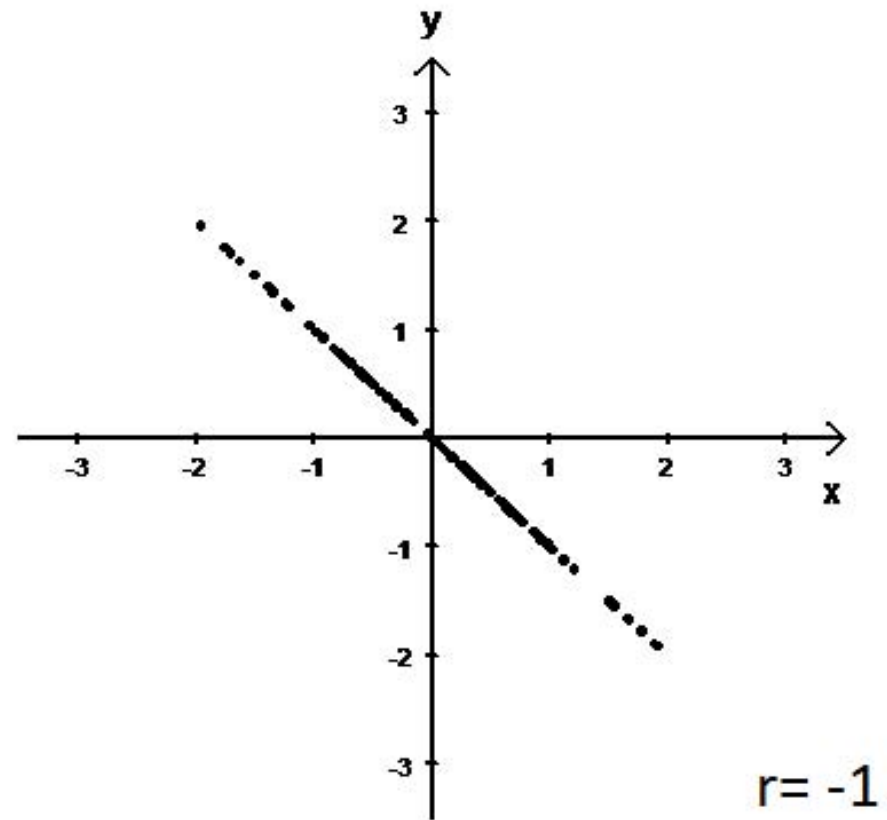
- $r_{xy} = +1$
- r_{xy} nahe bei 1
- $r_{xy} = -1$
- r_{xy} nahe bei -1
- $r_{xy} = 0$:
- r_{xy} nahe bei 0



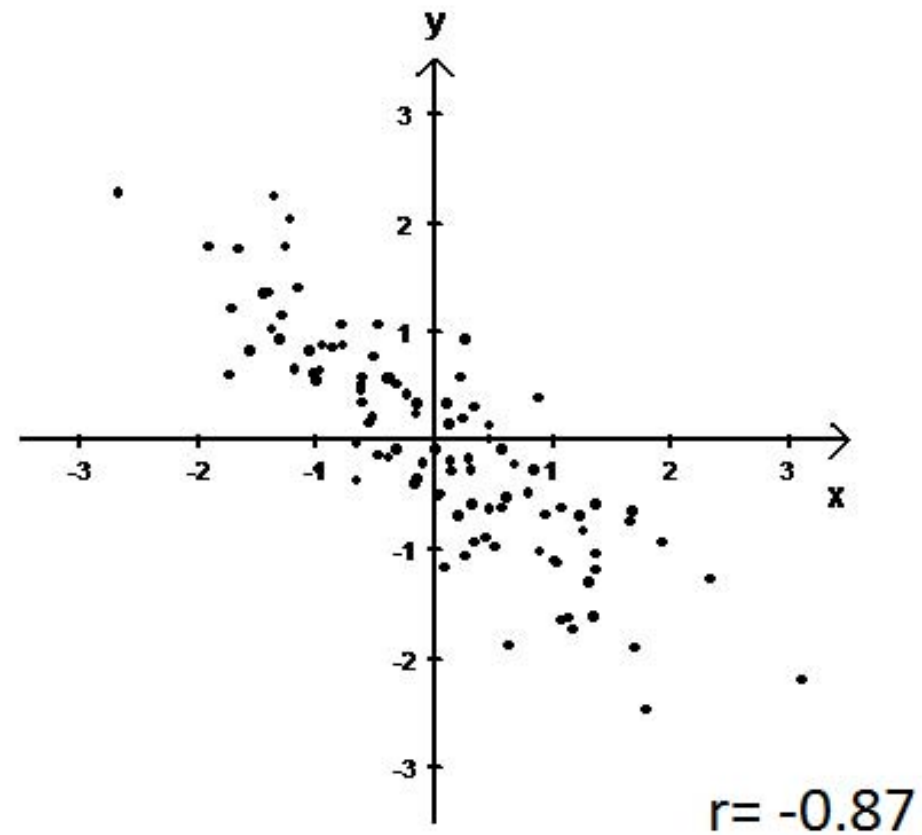
- $r_{xy} = +1$
- r_{xy} nahe bei 1
- $r_{xy} = -1$
- r_{xy} nahe bei -1
- $r_{xy} = 0$:
- r_{xy} nahe bei 0



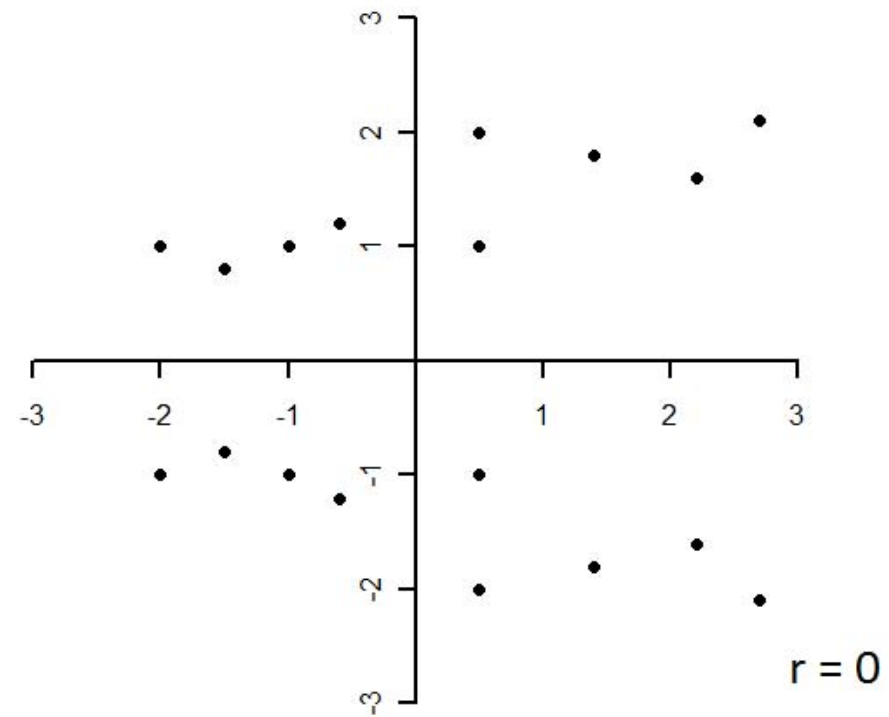
- $r_{xy} = +1$
- r_{xy} nahe bei 1
- $r_{xy} = -1$
- r_{xy} nahe bei -1
- $r_{xy} = 0$:
- r_{xy} nahe bei 0



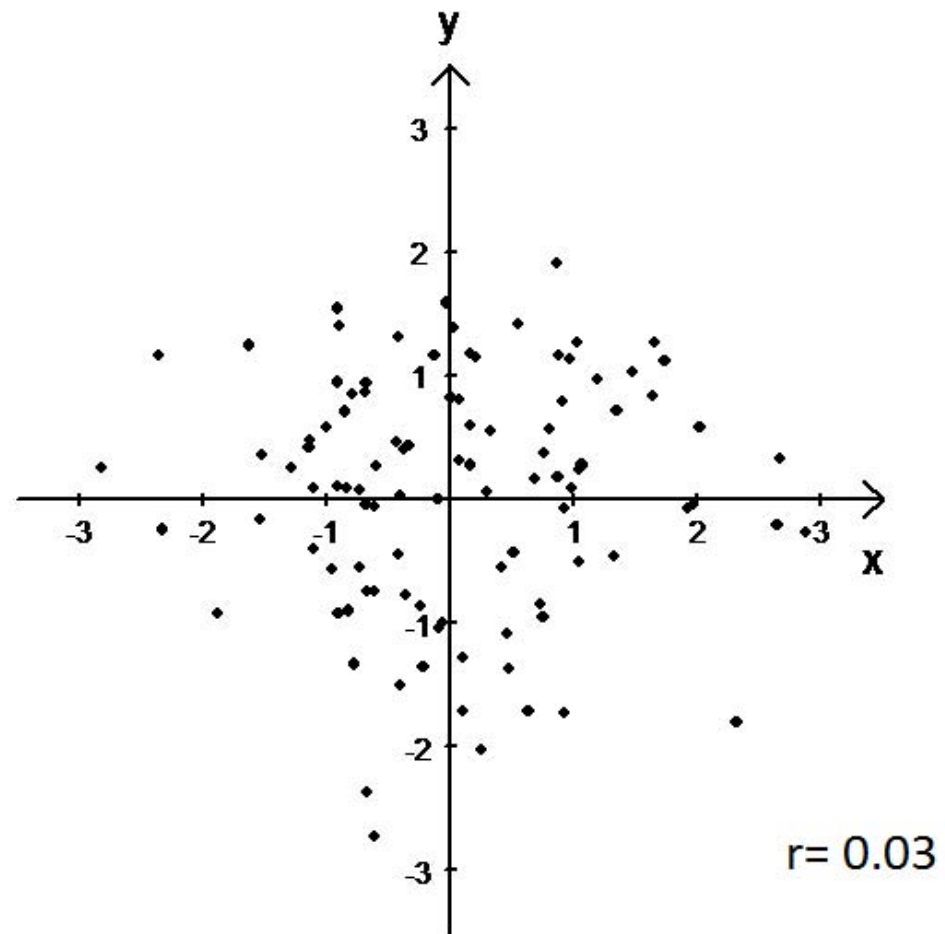
- $r_{xy} = +1$
- r_{xy} nahe bei 1
- $r_{xy} = -1$
- r_{xy} nahe bei -1
- $r_{xy} = 0$:
- r_{xy} nahe bei 0



- $r_{xy} = +1$
- r_{xy} nahe bei 1
- $r_{xy} = -1$
- r_{xy} nahe bei -1
- $r_{xy} = 0$:
- r_{xy} nahe bei 0



- $r_{xy} = +1$
- r_{xy} nahe bei 1
- $r_{xy} = -1$
- r_{xy} nahe bei -1
- $r_{xy} = 0$:
- r_{xy} nahe bei 0



Eigenschaften von Pearson's r_{xy}

- 1 $-1 \leq r_{xy} \leq +1$
- 2 $r_{xy} = +1$: alle Punkte liegen auf einer ansteigenden Gerade
- 3 r_{xy} nahe bei 1: x und y Werte wachsen gemeinsam
- 4 $r_{xy} = -1$: alle Punkte liegen auf einer abfallenden Gerade
- 5 r_{xy} nahe bei -1 : y fällt, wenn x wächst.
- 6 $r_{xy} = 0$: kein linearer (!) Zusammenhang
- 7 r_{xy} nahe bei 0 : keine gemeinsame Tendenz
- 8 $r_{xy} > 0$: positiver Zusammenhang
- 9 $r_{xy} < 0$: negativer Zusammenhang

Rang-Korrelationskoeffizient

Es gibt eine Vielzahl unterschiedlicher Korrelationskoeffizienten in Abhängigkeit von der Skalierung der Merkmale.

Mit dem Spearman-Rang-Korrelationskoeffizienten kann man den Zusammenhang zwischen 2 ordinalskalierten Merkmalen bestimmen. Wenn mindestens ein Merkmal ordinal skaliert ist, ist der Pearson-Korrelationskoeffizient r nicht anwendbar, da für ordinalskalierte Merkmale das arithmetische Mittel und Streuungen nicht zulässig sind. Man verwendet in diesem Fall den sogenannten Spearman-Korrelationskoeffizienten ρ .

Definition 3.3

Geordnete Statistik: Ordnen wir die Daten $x := (x_1, x_2, \dots, x_n)$ der Größe nach und fassen sie dann zu einem Vektor zusammen, so erhalten wir die s.g. *geordnete Statistik*: $(x_{(1)}, x_{(2)}, \dots, x_{(n)})$. Beispiel:

$$(x_1, x_2, x_3, x_4) = (45, 12, 30, 56)$$

$$(x_{(1)}, x_{(2)}, x_{(3)}, x_{(4)}) = (12, 30, 45, 56)$$

$$x_{(1)} = x_2, \quad x_{(2)} = x_3, \quad x_{(3)} = x_1, \quad x_{(4)} = x_4$$

Rang: Der Index j von $x_{(j)}$ gibt die Platzierung von $x_{(j)}$ in der geordneten Statistik an und wird als Rang bezeichnet:

$$r(x_1) = 3, \quad r(x_2) = 1, \quad r(x_3) = 2, \quad r(x_4) = 4$$

Spearman's ρ

Definition 3.4

Sei (x, y) wie bei Definition (3.1). Ordnet man die x- und y-Werte der Größe nach, so ergeben sich aus den ursprünglichen Daten (x_i, y_i) , $i = 1, 2, \dots, n$, neue Rangdaten $(r(x_i), r(y_i))$, $i = 1, 2, \dots, n$.

Der Spearmans Rang-Korrelationskoeffizient ρ wird nun berechnet durch den Pearson-Korrelationskoeffizienten, angewandt auf die Rangpaare $(r(x_i), r(y_i))$, $i = 1, 2, \dots, n$:

$$\rho_{xy} = \frac{s_{r_x r_y}}{s_{r_x} s_{r_y}}$$

Es gilt $-1 \leq \rho \leq 1$

Spearman's ρ : Beispiel

i	1	2	3	4
x_i	45	12	30	56
y_i	2	1.5	3.6	0.6
r_{x_i}	3	1	2	4
r_{y_i}	3	2	4	1

$\bar{r}_x, \bar{r}_y, s_{r_x}, s_{r_y}, s_{r_x r_y}$ berechnet man aus den Rang-daten.

cov()

```
var(iris[1:4]); cov(iris[1:4])
```

	Sepal_length	Sepal_width	Petal_length	Petal_width
Sepal_length	0.69	-0.04	1.27	0.52
Sepal_width	-0.04	0.19	-0.33	-0.12
Petal_length	1.27	-0.33	3.12	1.30
Petal_width	0.52	-0.12	1.30	0.58

cor()

```
cor(iris[1:4]) # method = "pearson"
```

	Sepal_length	Sepal_width	Petal_length	Petal_width
Sepal_length	1.00	-0.12	0.87	0.82
Sepal_width	-0.12	1.00	-0.43	-0.37
Petal_length	0.87	-0.43	1.00	0.96
Petal_width	0.82	-0.37	0.96	1.00

```
cor(iris[1:4],method="spearman")
```

	Sepal_length	Sepal_width	Petal_length	Petal_width
Sepal_length	1.00	-0.17	0.88	0.83
Sepal_width	-0.17	1.00	-0.31	-0.29
Petal_length	0.88	-0.31	1.00	0.94
Petal_width	0.83	-0.29	0.94	1.00

Kendall's τ

Hier wird die Abhängigkeit durch die Anzahl *konkordanter und diskordanter* Paare beschreiben.

Ein Paar $\{(x_i, y_i), (x_j, y_j)\}$ heißt konkordant (übereinstimmend), falls

$$(x_i < x_j \wedge y_i < y_j) \quad \vee \quad (x_i > x_j \wedge y_i > y_j)$$

sonst diskordant.

$$\tau = \frac{n_k - n_d}{\frac{n(n-1)}{2}}$$

wobei n_k und n_d Anzahl konkordanter- bzw. diskordanter Paare und $n(n-1)/2 = n_k + n_d$ sind.

Auch hier gilt $-1 \leq \tau \leq 1$.

cor()

```
cor(iris[1:4],method="kendall")
```

	Sepal_length	Sepal_width	Petal_length	Petal_width
Sepal_length	1.00	-0.08	0.72	0.66
Sepal_width	-0.08	1.00	-0.19	-0.16
Petal_length	0.72	-0.19	1.00	0.81
Petal_width	0.66	-0.16	0.81	1.00

Übung 3.26

Laden Sie den Datensatz *trees* aus dem Paket *datasets* und Bestimmen Sie die Kovarianz- und die Korrelationsmatrizen für die Variablen.

Erstellen Sie die Streudiagramm-Matrix für die Merkmale.

Verwenden Sie jetzt die Funktion *pairs()* statt *plot()* mit der Option *panel=panel.smooth* für die Darstellung des Streudiagramms und interpretieren Sie die Ergebnisse.

Gliederung I

Organisatorisches

- | | |
|--|-----|
| 1. Einstieg und Grundlegendes zu  | 42 |
| 2. Datentypen und Datenimport | 92 |
| 3. Deskriptive Statistik mit <i>R</i> | 193 |
| 4. Verteilungen & Zufallszahlen in  | 349 |
| 5. Schliessende Statistik: Testen und Schätzen | 441 |

4. Verteilungen & Zufallszahlen in

- ▶ diskrete Verteilungen
 - Binomialverteilung
 - Multinomialverteilung
 - Poisson-Verteilung
- ▶ stetige Verteilungen
 - Normalverteilung
 - Chi-Quadrat-Verteilung
 - t-Verteilung
 - F-Verteilung
 - Gleichverteilung (*uniforme Verteilung*)

- Funktionen für die *Dichten* (**d**ensity), *Verteilungsfunktionen* (**p**robability), *Quantile* (**q**uantile) und Generatoren von (*Pseudo*-)Zufallszahlen (**r**andom numbers) häufig gebrauchter Verteilungen sind implementiert.
- Die entsprechenden Funktionsnamen bestehen aus:
Funktionskürzel (d,p,q,r) + **Verteilungskürzel** (norm, binom,..)
- Beispiel: **dnorm**:
d für **density**, **norm** für Normal distribution

```
dnorm(0.45) # liefert die Dichte von N(0,1)
            # an der Stelle x = 0.45.
0.360527
```

Verteilungen & Zufallszahlen

<i>Präfix</i>	<i>Beschreibung</i>
d	Dichte (<i>density</i>)
p	Verteilungsfunktion (<i>probability</i>)
q	Quantilsfunktion (<i>quantile</i>)
r	Zufallszahl (<i>random number</i>)

<i>Suffix</i>	<i>Verteilung</i>
norm	Normal-
lnorm	Lognormal
t	t-
chisq	χ^2 -
f	f-
exp	Exponential-
unif	Gleich-
beta	Beta-
cauchy	Cauchy-
binom	Binomial-
multinom	Multinomial-
pois	Poisson-
geom	Geometrische-

Normalverteilung

<code>dnorm(2, m=2, sd=3)</code>	0.1329808	
<code>dnorm(-1, 0, 1)</code>	0.2419707	
<code>dnorm(-5, 4, 2)</code>	7.991871e-06	
<code>pnorm(0, m=0, sd=1)</code>	0.5	
<code>pnorm(2, m=1, sd=2)</code>	0.6914625	
<code>qnorm(0.95, m=0, sd=1)</code>	1.644854	
<code>qnorm(0.001, m=10, sd=5)</code>	-5.451162	
<code>rnorm(2, m=1, sd=1)</code>	0.02492578	1.41583180

.....

?dnorm

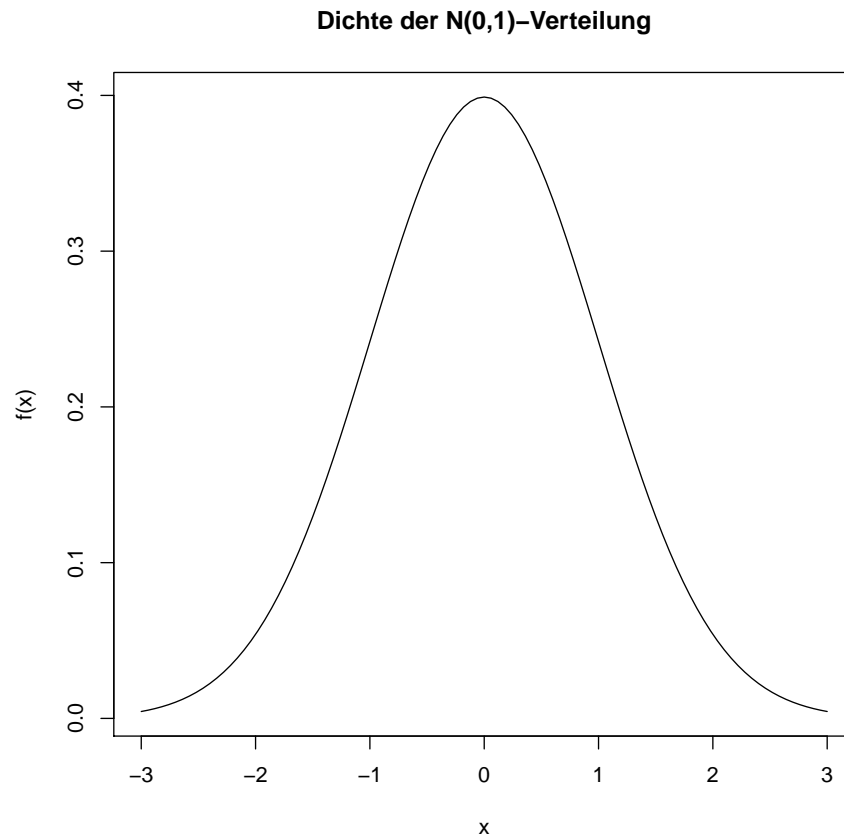
```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE)
rnorm(n, mean = 0, sd = 1)
# mean=m ; sd=s
```

Dichte der Normalverteilung: curve()

curve

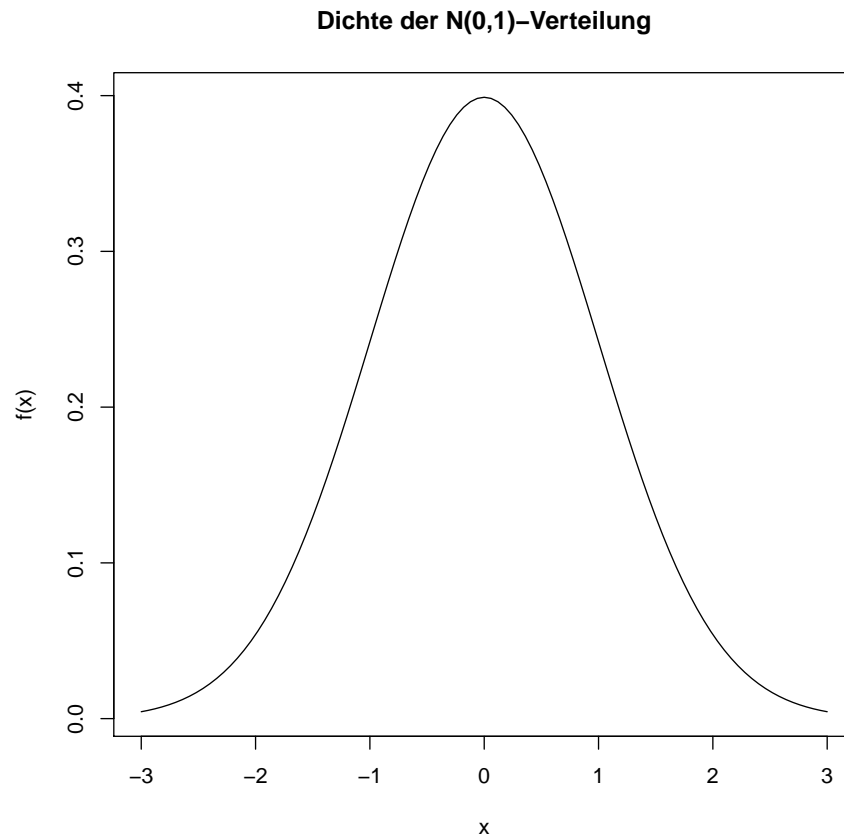
```
curve(expr, from = NULL, to = NULL, n = 101,  
      add = FALSE, type = "l")
```

```
curve(dnorm(x), from=-3, to=3,  
      main="Dichte_der_N(0,1)-Verteilung", ylab="f(x)")
```



Dichte der Normalverteilung: plot()

```
x <- seq(-3, 3, 0.01)
y <- dnorm(x)
plot(x, y, type="l",
      main="Dichte der N(0,1)-Verteilung", ylab="f(x)")
```



Dichte der Normalverteilung: plot()

`plot(function(x))`
`?plot.function`

Alternativ:

```
plot(dnorm, -3, 3,  
     main="Dichte_der_N(0,1)-Verteilung", ylab="f(x)")
```

Anderes Bsp.:

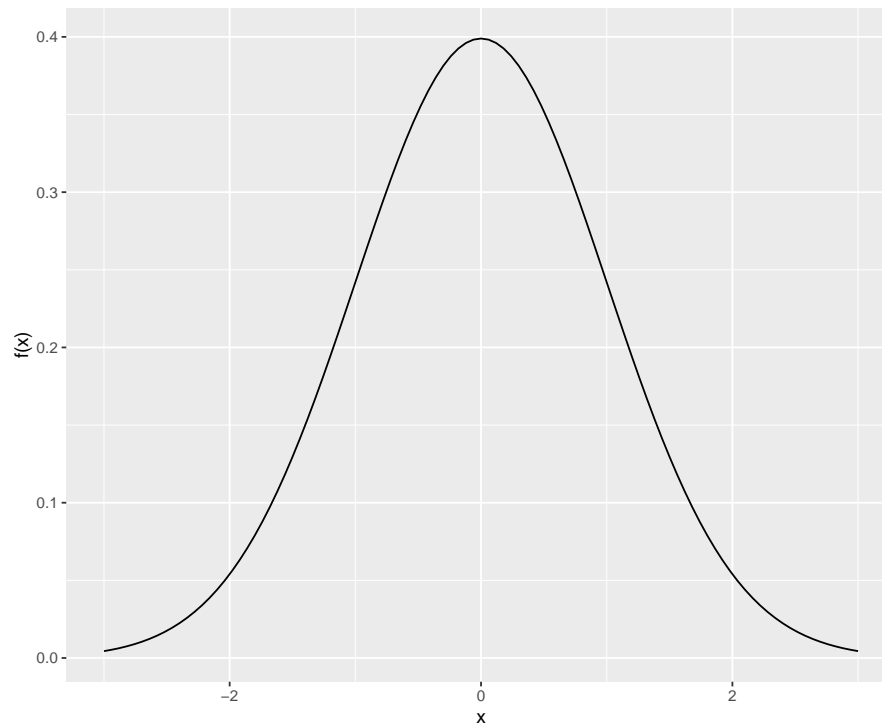
```
fun2 <- function(x) sin(x)/x  
plot(fun2, 0, 50)
```

Also:

```
curve(dnorm(x), -3, 3, main="N(0,1)", ylab="f(x)") # oder  
plot(dnorm, xlim=c(-3,3), main="N(0,1)", ylab="f(x)")
```

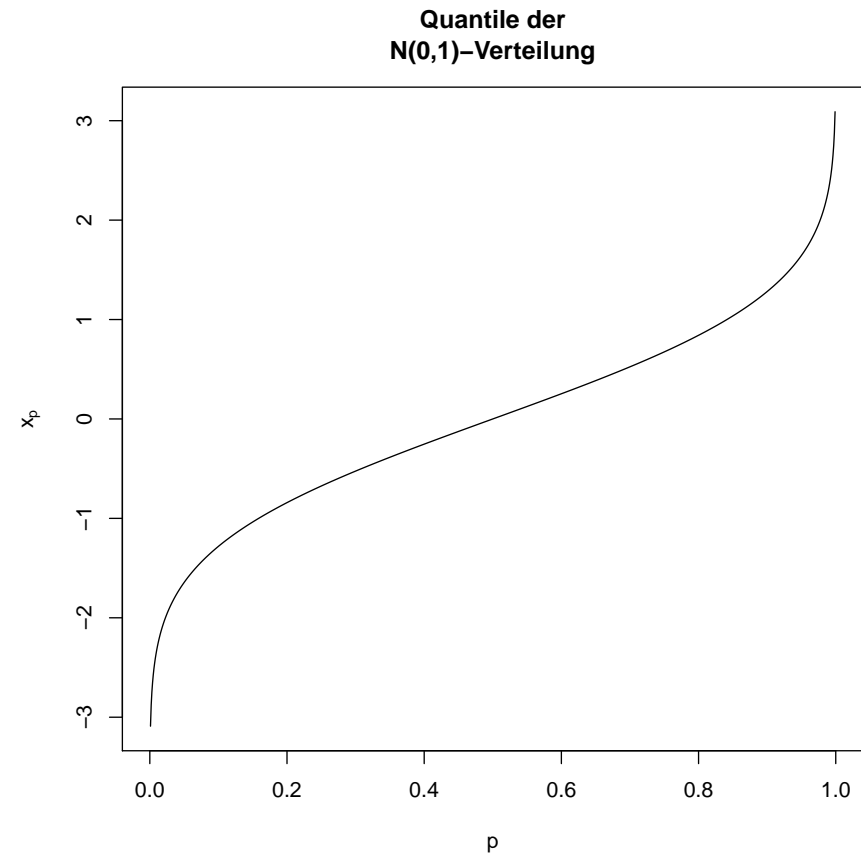
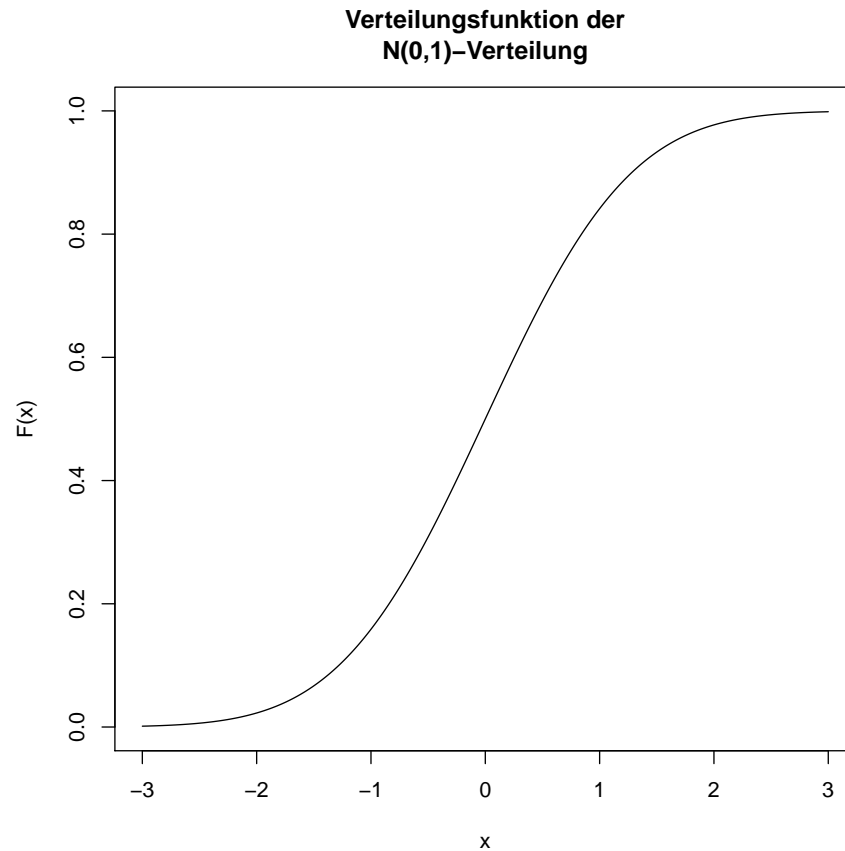
Dichte der Normalverteilung mit ggplot2

```
library(ggplot2)
ggplot(data = data.frame(x = c(-3, 3)), aes(x)) +
  stat_function(fun = dnorm, n = 101,
    args = list(mean = 0, sd = 1)) +
  ylab("f(x)")
```



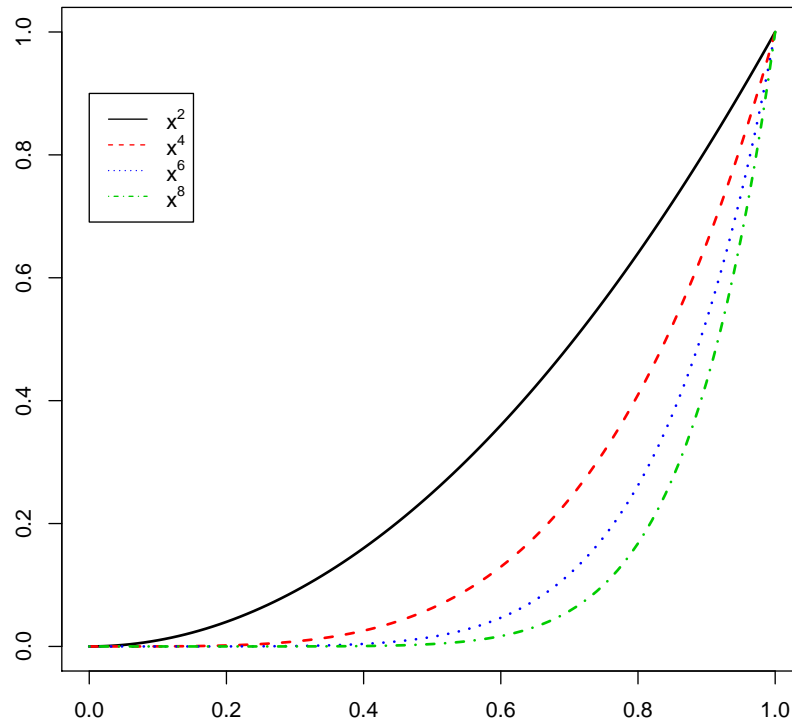
Übung 4.1

Stellen Sie die folgenden Grafiken mit R dar.

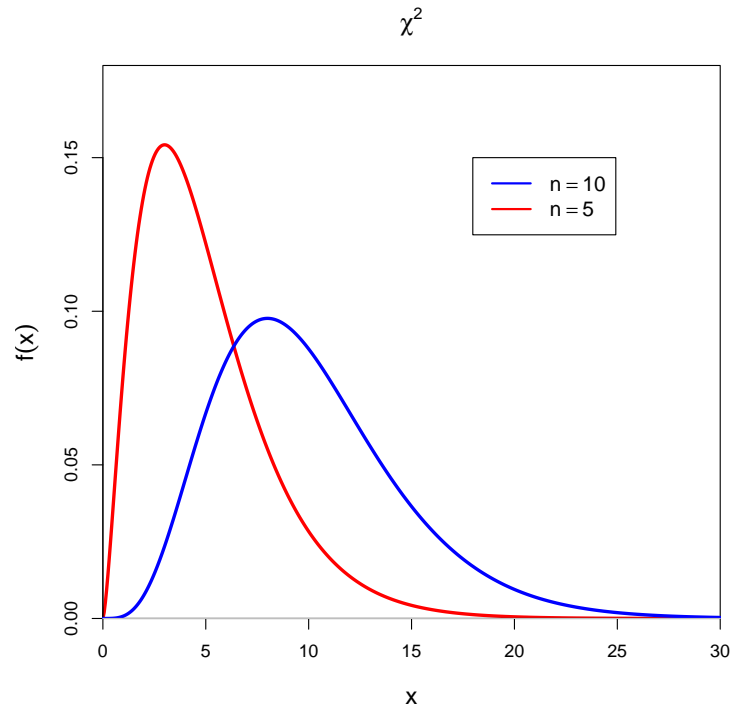


Funktionen zeichnen mit curve(): Beispiel

```
curve(x^2, from=0, to=1, col="black", lty=1, lwd=2, ylab="")
curve(x^4, from=0, to=1, col="red", lty=2, lwd=2, add=TRUE)
curve(x^6, from=0, to=1, col="blue", lty=3, lwd=2, add=TRUE)
curve(x^8, from=0, to=1, col="green3", lty=4, lwd=2, add=TRUE)
legend(0, 0.9,
      legend=c(expression(x^2), expression(x^4),
                expression(x^6), expression(x^8)),
      col = c("black", "red", "blue", "green3"), lty = 1:4)
```



Weitere stetige Verteilungen: χ_n^2



Definition 4.1

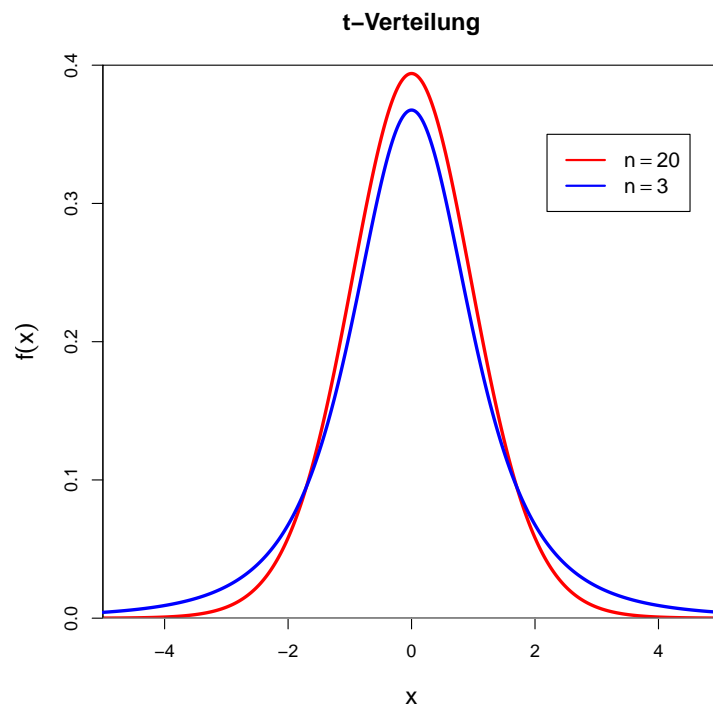
Seien $X_i \sim N(0,1)$, für $i = 1, 2, \dots, n$ unabhängige ZV. Dann ist

$$X = \sum_{i=1}^n X_i^2$$

eine χ^2 -verteilte ZV mit n Freiheitsgraden.

$$f(x) = \frac{1}{2^{n/2} \Gamma(n/2)} x^{n/2-1} e^{-x/2} \quad \text{für } x > 0$$

Weitere stetige Verteilungen: t_n



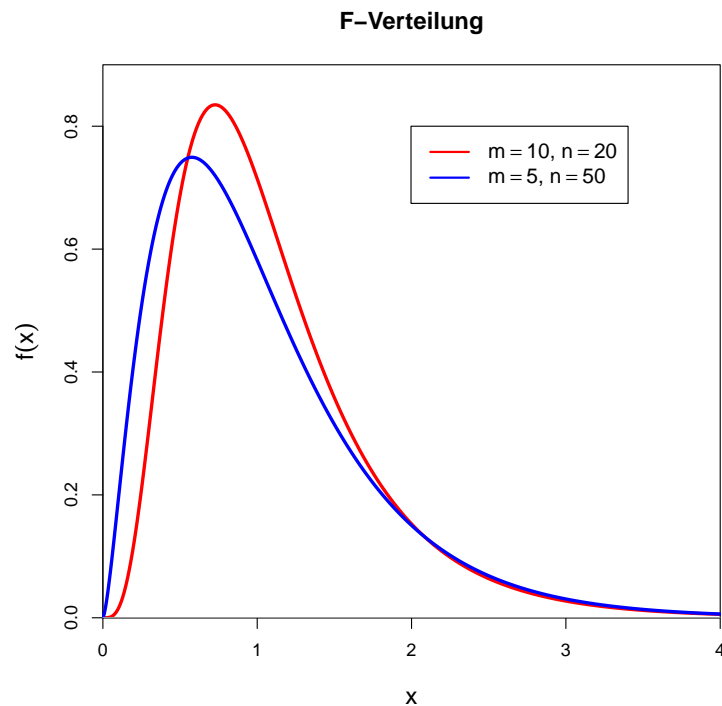
Definition 4.2

Seien $X \sim N(0, 1)$ und $Y \sim \chi_n^2$ unabhängige ZV. Dann ist

$$Z = \frac{X}{\sqrt{\frac{Y}{n}}}$$

eine t -verteilte ZV mit n Freiheitsgraden.

Weitere stetige Verteilungen: $F_{m,n}$



Definition 4.3

Seien $X \sim \chi_n^2$ und $Y \sim \chi_m^2$ unabhängige ZV. Dann ist

$$Z = \frac{\frac{X}{n}}{\frac{Y}{m}}$$

eine F -verteilte ZV mit Freiheitsgraden m und n .

Beispiel: t -, F - und χ^2 - Verteilungen

p-Quantile der χ^2_{10} -, t_4 - und $F_{8,6}$ -Verteilungen für
 $p = 0.01, 0.02, 0.03, \dots, 0.99$ berechnen:

Beispiel: t -, F - und χ^2 - Verteilungen

p-Quantile der χ^2_{10} -, t_4 - und $F_{8,6}$ -Verteilungen für
 $p = 0.01, 0.02, 0.03, \dots, 0.99$ berechnen:

```
x <- seq(0.01, 0.99, 0.01)
y1 <- qchisq(x, df=10)
y2 <- qt(x, df=4)
y3 <- qf(x, df1=8, df2=6)
z <- data.frame(Quantile=x, Chi.10=y1, t.4=y2, f.8.6=y3)
```

Beispiel: t -, F - und χ^2 - Verteilungen II

Es sei $X \sim t_5$. Wie bestimmt man:

- i) $P(X \leq 1)$ und $P(X \geq -0.5)$,
- ii) $X_{0.1}$ und $X_{0.9}$?

Beispiel: t -, F - und χ^2 - Verteilungen II

Es sei $X \sim t_5$. Wie bestimmt man:

i) $P(X \leq 1)$ und $P(X \geq -0.5)$,

ii) $X_{0.1}$ und $X_{0.9}$?

```
pt(1, df=5)
```

```
1-pt(-0.5, df=5); pt(-0.5, df=5, lower.tail = F)
```

```
qt(c(0.1, 0.9), df=5)
```

Übung 4.2

Stellen Sie die Dichtefunktionen der t -Verteilungen mit 3, 5, 10, 30 und 50 Freiheitsgraden und die Dichtefunktion einer Standard-Normalverteilung gemeinsam in einer Abbildung dar. Verwenden Sie verschiedene Farben oder/und Linientypen.

Gaussian mixture distribution – Präsenzaufg. 1

Übung 4.3

- a. Schreiben Sie eine Funktion (mit den Parametern $p, \mu_1, \mu_2, \sigma_1, \sigma_2, n$), die Zufallszahlen aus einer *gaussian mixture distribution* mit 2 Komponenten generiert.

Die Dichtefunktion dieser Verteilung ist:

$$f(x) = pf_1(x, \mu_1, \sigma_1) + (1 - p)f_2(x, \mu_2, \sigma_2).$$

Hinweis: verwenden Sie die Funktionen `rbinom()` und `rnorm()`.

- b. Ziehen Sie $n = 1000$ Stichproben aus

$$X \sim 0.2N(160, 10) + 0.8N(180, 10)$$

und stellen Sie die Verteilung der Daten durch eine Kerndichteschätzung dar. Fügen Sie der Grafik die theoretische Dichtefunktion hinzu.

Diskrete Verteilungen: Bernoulli-

Besitzt ein Zufallsexperiment nur zwei mögliche Ausgänge, so spricht man von einem Bernoulli-Experiment. Dieses wird i.A. mit einer Zufallsvariable X beschrieben, welche die Werte 0 oder 1 annimmt.

Definition 4.4

Eine diskrete Zufallsvariable X ist Bernoulli-verteilt mit dem Parameter p , wenn für deren Einzelwahrscheinlichkeiten gilt:

$$P(X = 1) = p \quad \text{und} \quad P(X = 0) = 1 - p$$

Diskrete Verteilungen: Binomial-

Definition 4.5

Sei X die Anzahl der Erfolge n unabhängiger Bernoulli-Experimente mit Erfolgswahrscheinlichkeit p . Dann ist X binomialverteilt und es gilt

$$P(\{X = k\}) = \binom{n}{k} p^k (1 - p)^{(n-k)}$$

für $k = 0, 1, \dots, n$

Schreibweise: $X \sim B(n; p)$

$$P(\{X = k\}) = \binom{n}{k} p^k (1 - p)^{(n-k)} : \quad \text{W-Funktion (W-Verteilung)}$$

$$P(\{X \leq k\}) = \sum_{i=0}^k \binom{n}{i} p^i (1 - p)^{(n-i)} : \quad \text{Verteilungsfunktion}$$

Binomialverteilung mit

```
dbinom(x=5, size=10, prob=.4)           0.2006581
pbinom(q=5, size=10, prob=.4, lower.tail = T) 0.8337614
qbinom(p=.95, size=10, prob=.4, lower.tail = T) 7
rbinom(n=3, size=10, prob=.5)             4    6    7
.....
```

Binomialverteilung mit

```
dbinom(x=5, size=10, prob=.4)           0.2006581
pbinom(q=5, size=10, prob=.4, lower.tail = T) 0.8337614
qbinom(p=.95, size=10, prob=.4, lower.tail = T) 7
rbinom(n=3, size=10, prob=.5)             4    6    7
```

.....

Es sei $X \sim B(n = 10, p = 0.5)$.

1. Wir stellen die W-funktion von X dar.

Binomialverteilung mit

```
dbinom(x=5, size=10, prob=.4)           0.2006581
pbinom(q=5, size=10, prob=.4, lower.tail = T) 0.8337614
qbinom(p=.95, size=10, prob=.4, lower.tail = T) 7
rbinom(n=3, size=10, prob=.5)           4    6    7
```

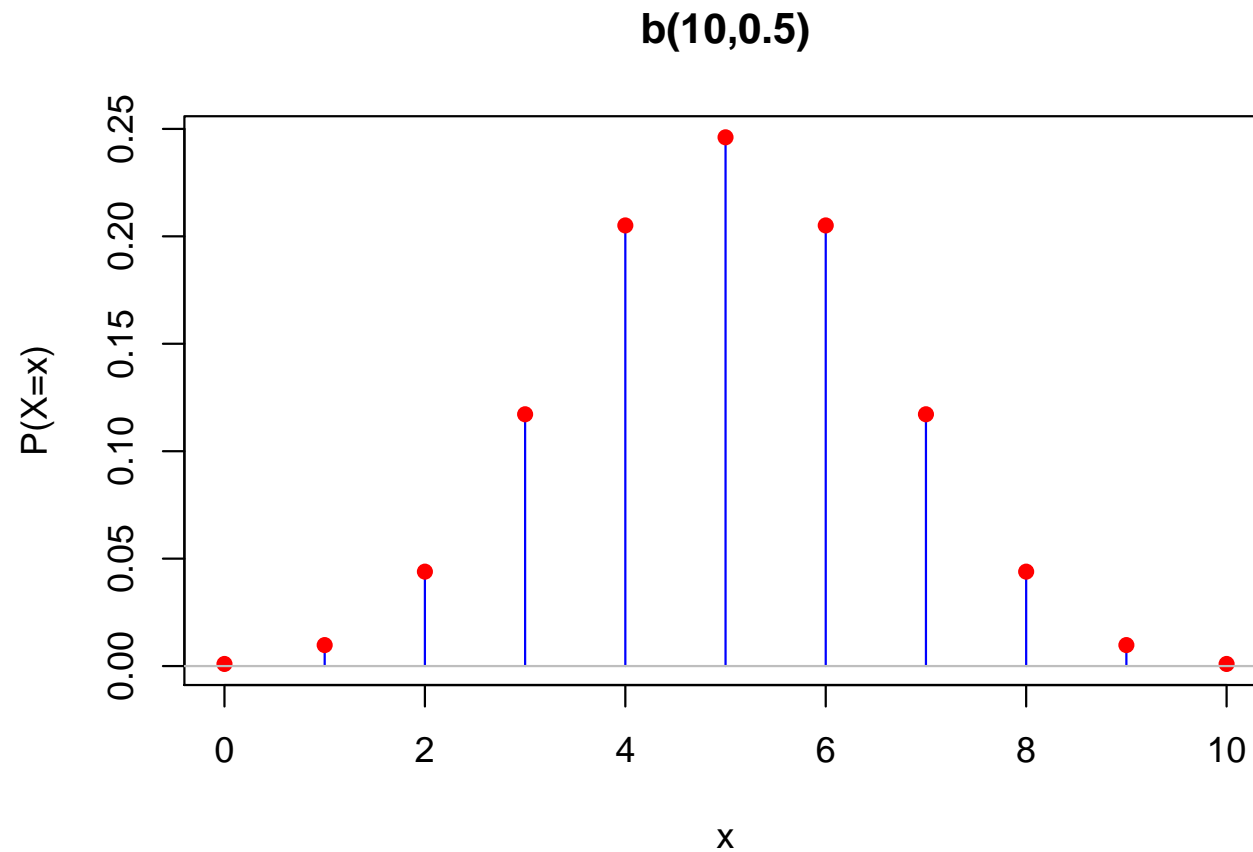
.....

Es sei $X \sim B(n = 10, p = 0.5)$.

1. Wir stellen die W-funktion von X dar.

```
x <- 0:10
plot(x, dbinom(x, size=10, prob=0.5), xlab="x",
     ylab="P(X=x)", main="b(10,0.5)", type="h", col="blue")
points(x, dbinom(x, size=10, prob=0.5),
       pch=16, col="red", lwd=2)
abline(h=0, col="gray")
```

Binomialverteilung mit II



2. Simulieren wir 1000 Samples von $B(n = 10, p = 0.5)$ -verteilten Zufallsvariablen.

2. Simulieren wir 1000 Samples von $B(n = 10, p = 0.5)$ -verteilten Zufallsvariablen.

```
data1 <- rbinom(n=1000, size=10, prob=0.5)
```


2. Simulieren wir 1000 Samples von $B(n = 10, p = 0.5)$ -verteilten Zufallsvariablen.

```
data1 <- rbinom(n=1000, size=10, prob=0.5)
```

3. Berechnen wir den Mittelwert und die Varianz von Daten und vergleichen wir diese mit $E(X)$ und $Var(X)$.

2. Simulieren wir 1000 Samples von $B(n = 10, p = 0.5)$ -verteilten Zufallsvariablen.

```
data1 <- rbinom(n=1000, size=10, prob=0.5)
```

3. Berechnen wir den Mittelwert und die Varianz von Daten und vergleichen wir diese mit $E(X)$ und $Var(X)$.

```
mean(data1) # 10*0.5  
var(data1)  # 10*0.5*(1-0.5)
```

set.seed()

```
rbinom(n=2, size=10, prob=0.5)
```

```
6 5
```

```
rbinom(n=2, size=10, prob=0.5)
```

```
5 4
```

set.seed()

```
rbinom(n=2, size=10, prob=0.5)  
6 5
```

```
rbinom(n=2, size=10, prob=0.5)  
5 4
```

Damit die Ergebnisse reproduzierbar werden, verwenden wir die Funktion *set.seed()*.

```
set.seed(123)  
rbinom(n=2, size=10, prob=0.5)  
4 6
```

set.seed()

```
rbinom(n=2, size=10, prob=0.5)  
6 5
```

```
rbinom(n=2, size=10, prob=0.5)  
5 4
```

Damit die Ergebnisse reproduzierbar werden, verwenden wir die Funktion *set.seed()*.

```
set.seed(123)  
rbinom(n=2, size=10, prob=0.5)  
4 6
```

```
set.seed(123)  
rbinom(n=2, size=10, prob=0.5)  
4 6
```

set.seed()

```
rbinom(n=2, size=10, prob=0.5)  
6 5
```

```
rbinom(n=2, size=10, prob=0.5)  
5 4
```

Damit die Ergebnisse reproduzierbar werden, verwenden wir die Funktion *set.seed()*.

```
set.seed(123)  
rbinom(n=2, size=10, prob=0.5)  
4 6
```

```
set.seed(123)  
rbinom(n=2, size=10, prob=0.5)  
4 6
```

```
rbinom(n=2, size=10, prob=0.5)  
5 7
```

set.seed()

```
n1 <- 123
set.seed(n1)
x1 <- runif(10)
set.seed(n1)
x2 <- runif(10) ; x1-x2 ; identical(x1, x2)
```

set.seed: Vorsicht bei Simulationen

```
N <- 10; M <- 5
X <- matrix(nrow=N, ncol=M)
for (i in 1:N){
  set.seed(4)
  X[i,] <- runif(M)
} ; X
apply(X, 2, var)
```


set.seed: Vorsicht bei Simulationen

```
N <- 10; M <- 5
X <- matrix(nrow=N, ncol=M)
for (i in 1:N){
  set.seed(4)
  X[i,] <- runif(M)
} ; X
apply(X, 2, var)

for (i in 1:N){
  set.seed(i+3)
  X[i,] <- runif(M)
} ; X
apply(X, 2, var)
```

set.seed: Vorsicht bei Simulationen

```
N <- 10; M <- 5
X <- matrix(nrow=N, ncol=M)
for (i in 1:N){
  set.seed(4)
  X[i,] <- runif(M)
} ; X
apply(X, 2, var)
```

```
for (i in 1:N){
  set.seed(i+3)
  X[i,] <- runif(M)
} ; X
apply(X, 2, var)
```

```
set.seed(4)
X <- matrix(nrow=N, ncol=M)
for (i in 1:N){
  X[i,] <- runif(M)
}; X
apply(X, 2, var)
```

sample()

sample(x, size, replace=FALSE, prob=NULL)

Zieht eine Stichprobe der Größe **size** aus dem Vektor **x**
ohne Zurücklegen (**replace=FALSE**).

sample()

sample(x, size, replace=FALSE, prob=NULL)

Zieht eine Stichprobe der Größe **size** aus dem Vektor **x**
ohne Zurücklegen (**replace=FALSE**).

```
data1<-sample(x=0:10, size=2); data1  
7      1
```

sample()

sample(x,size,replace=FALSE,prob=NULL)

Zieht eine Stichprobe der Größe **size** aus dem Vektor **x**
ohne Zurücklegen (**replace=FALSE**).

```
data1<-sample(x=0:10, size=2); data1  
7 1
```

Jetzt mit *set.seed()*:

```
set.seed(1001)  
data1<-sample(x=0:10, size=2); data1  
10 4
```

```
set.seed(1001)  
data1<-sample(x=0:10, size=2); data1  
10 4
```

sample()

sample(x,size,replace=FALSE,prob=NULL)

Zieht eine Stichprobe der Größe **size** aus dem Vektor **x**
ohne Zurücklegen (**replace=FALSE**).

```
data1<-sample(x=0:10, size=2); data1  
7 1
```

Jetzt mit *set.seed()*:

```
set.seed(1001)
```

```
data1<-sample(x=0:10, size=2); data1  
10 4
```

```
set.seed(1001)
```

```
data1<-sample(x=0:10, size=2); data1  
10 4
```

sample(x,size,replace=TRUE,prob=NULL)

Zieht eine Stichprobe der Größe **size** aus dem Vektor **x**
mit Zurücklegen (**replace=TRUE**).

```
data1<-sample(x=0:1, size=7, replace=TRUE); data1  
0 0 1 0 1 1 1
```

Beispiel: rbinom(), sample()

Wir würfeln 1000 Mal und bestimmen wie oft die *Sechs* gewürfelt wurde.

```
sum(rbinom(n=1000, size=1, prob=1/6)) # oder  
rbinom(n=1, size=1000, prob=1/6) # oder  
data <- sample(x=1:6, size=1000, replace=TRUE)  
sum(data==6)
```

Diskrete Verteilungen: *rdiscrete*

```
library(class); library(e1071)
rdiscrete(n, probs, values = 1:length(probs))
```

Zieht eine Stichprobe der Größe **n** aus dem Vektor **values** mit gegebenen Wahrscheinlichkeiten **probs**.

```
set.seed(1234)
rdiscrete(5, c(0.2, 0.8))
2 2 2 2 1
```

```
set.seed(1234)
rdiscrete(5, c(0.2, 0.8), values=c("Med.A", "Med.B"))
"Med.B" "Med.B" "Med.B" "Med.B" "Med.A"
```

```
set.seed(1234) # äquivalent:
rdiscrete(5, c(1,4), values=c("Med.A", "Med.B"))

"Med.B" "Med.B" "Med.B" "Med.B" "Med.A"
```


Diskrete Verteilungen: *rdiscrete*

```
library(class); library(e1071)
rdiscrete(n, probs, values = 1:length(probs))
```

Zieht eine Stichprobe der Größe **n** aus dem Vektor **values** mit gegebenen Wahrscheinlichkeiten **probs**.

```
set.seed(1234)
rdiscrete(5, c(0.2, 0.8))
2 2 2 2 1
```

```
set.seed(1234)
rdiscrete(5, c(0.2, 0.8), values=c("Med.A", "Med.B"))
"Med.B" "Med.B" "Med.B" "Med.B" "Med.A"
```

```
set.seed(1234) # äquivalent:
rdiscrete(5, c(1, 4), values=c("Med.A", "Med.B"))

"Med.B" "Med.B" "Med.B" "Med.B" "Med.A"
```

<pre>rdiscrete(n=5, probs=c(0.2, 0.8), values=c("Med.A", "Med.B")) ≡ sample(x=c("Med.A", "Med.B"), size=5, prob=c(.2,.8), replace=T)</pre>
--

Siehe *ddiscrete*, *pdiscrete*, *qdiscrete*.

Beispiel: Lotto Spielen (6 aus 49)

```
mein.tip <- c(10,15,20,25,30,35)
gew.woche <- sort(sample(1:49,6))
b <- intersect(mein.tip,gew.woche)
mein.tip
gew.woche
b
length(b)
```

Übung 4.4

1. Angenommen man spielt 1000 Wochen Lotto 6 aus 49.
Schreiben Sie eine Funktion, die berechnet, wie oft man 3 (4, 5 oder 6) *richtige* Zahlen getippt hat.
2. Angenommen Sie entscheiden sich jede Woche einmal zu spielen, bis erstmals 3 (4, 5 oder 6) *Richtige* auftreten, aber maximal 10000 mal.
Schreiben Sie eine Funktion, die dieses Spiel ermöglicht und ausgibt wann (wenn überhaupt) das Spiel gewonnen wurde.
3. Wie 2., aber diesmal geben Sie 100 Jahre lang pro Woche 4 Tipps ab, bis erstmals 5 Richtige auftreten, wobei pro Jahr 52 Mal gespielt wird.

Grenzwertsatz von Moivre-Laplace

$$X_1, X_2, \dots, X_n \sim \text{Bern}(p), \quad E(X_i) = p, \text{Var}(X_i) = p(1 - p) \Rightarrow$$

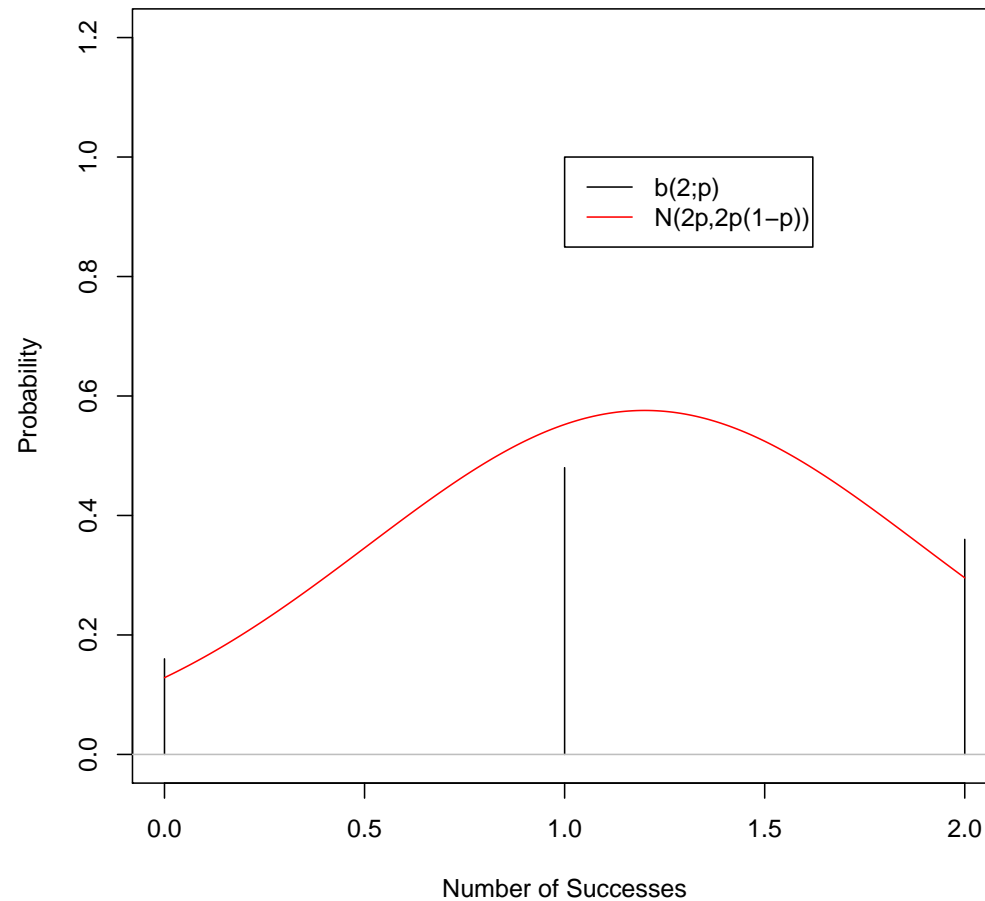
$$S_n := \sum_{i=1}^n X_i \sim B(n; p), \quad E(S_n) = np, \text{Var}(S_n) = np(1 - p)$$

Der Grenzwertsatz von Moivre-Laplace besagt, dass sich die Verteilung der Zufallsvariablen S_n , für **große** n , durch die normalverteilte Zufallsvariable Y approximieren lässt:

$$Y \sim N\left(\mu = E(S_n), \sigma^2 = \text{Var}(S_n)\right) = N\left(np, np(1 - p)\right)$$

Wir betrachten diese für

$p = 0.6$ und $n = 2, 5, 10, 20, 50, 100, 1000$.



$$X_1, X_2 \sim \text{Bern}(p)$$

$$S_2 \sim B(2; p)$$

$$E(S_2) = 2p$$

$$\text{Var}(S_2) = 2p(1 - p)$$

$$X_1, \dots, X_5 \sim \text{Bern}(p)$$

$$S_5 \sim B(5; p)$$

$$E(S_5) = 5p$$

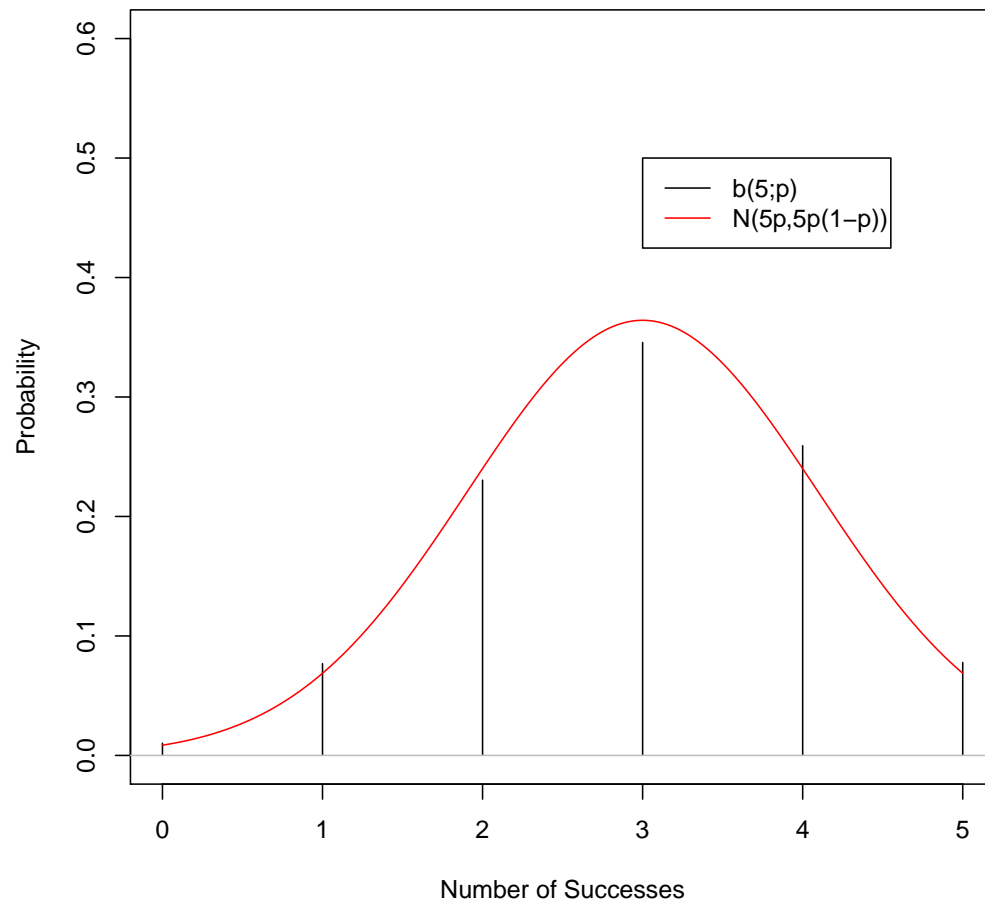
$$\text{Var}(S_5) = 5p(1 - p)$$

$$X_1, \dots, X_{10} \sim \text{Bern}(p)$$

$$S_{10} \sim B(10; p)$$

$$E(S_{10}) = 10p$$

$$\text{Var}(S_{10}) = 10p(1 - p)$$



$$X_1, X_2 \sim \text{Bern}(p)$$

$$S_2 \sim B(2; p)$$

$$E(S_2) = 2p$$

$$\text{Var}(S_2) = 2p(1 - p)$$

$$X_1, \dots, X_5 \sim \text{Bern}(p)$$

$$S_5 \sim B(5; p)$$

$$E(S_5) = 5p$$

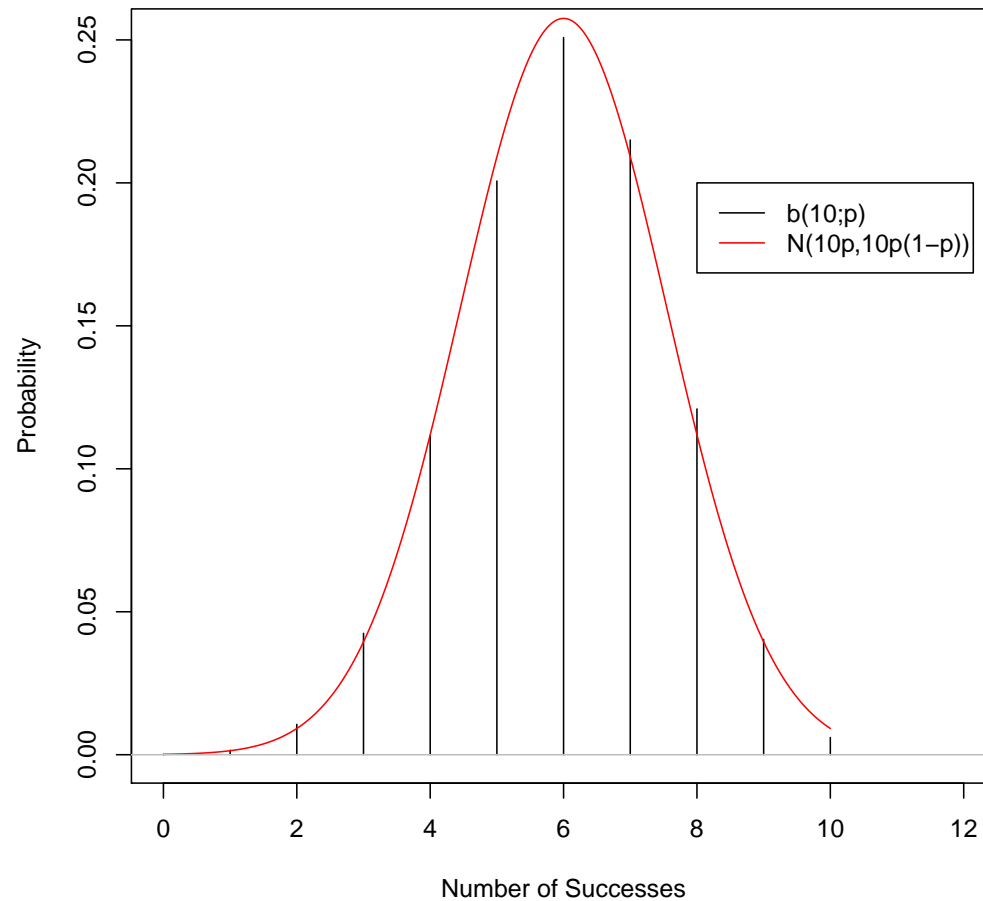
$$\text{Var}(S_5) = 5p(1 - p)$$

$$X_1, \dots, X_{10} \sim \text{Bern}(p)$$

$$S_{10} \sim B(10; p)$$

$$E(S_{10}) = 10p$$

$$\text{Var}(S_{10}) = 10p(1 - p)$$



$$X_1, X_2 \sim \text{Bern}(p)$$

$$S_2 \sim B(2; p)$$

$$E(S_2) = 2p$$

$$\text{Var}(S_2) = 2p(1 - p)$$

$$X_1, \dots, X_5 \sim \text{Bern}(p)$$

$$S_5 \sim B(5; p)$$

$$E(S_5) = 5p$$

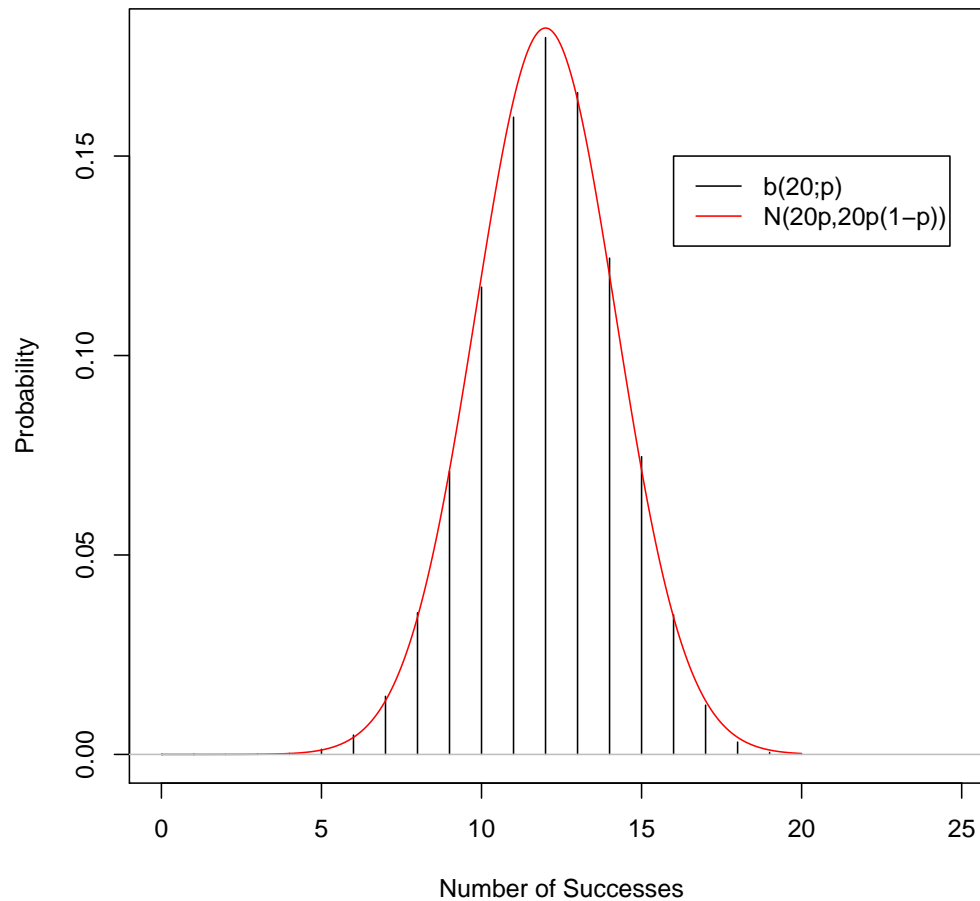
$$\text{Var}(S_5) = 5p(1 - p)$$

$$X_1, \dots, X_{10} \sim \text{Bern}(p)$$

$$S_{10} \sim B(10; p)$$

$$E(S_{10}) = 10p$$

$$\text{Var}(S_{10}) = 10p(1 - p)$$



$$X_1, X_2 \sim \text{Bern}(p)$$

$$S_2 \sim B(2; p)$$

$$E(S_2) = 2p$$

$$\text{Var}(S_2) = 2p(1 - p)$$

$$X_1, \dots, X_5 \sim \text{Bern}(p)$$

$$S_5 \sim B(5; p)$$

$$E(S_5) = 5p$$

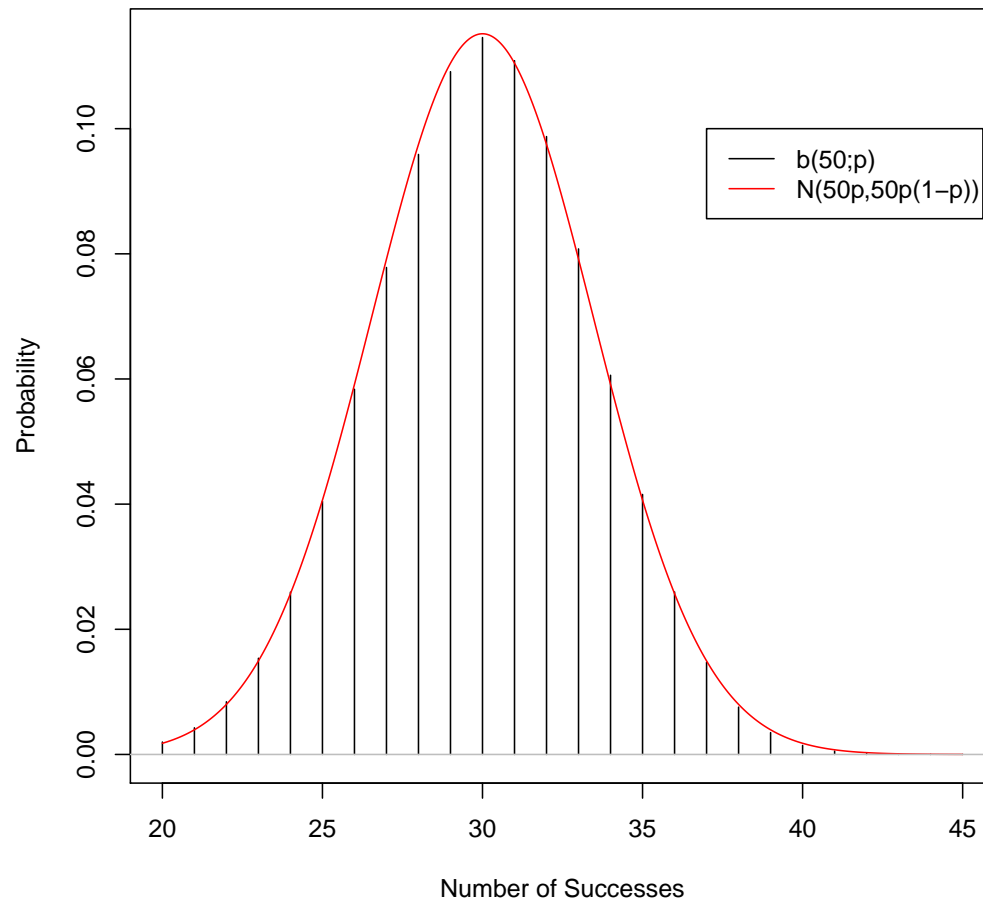
$$\text{Var}(S_5) = 5p(1 - p)$$

$$X_1, \dots, X_{10} \sim \text{Bern}(p)$$

$$S_{10} \sim B(10; p)$$

$$E(S_{10}) = 10p$$

$$\text{Var}(S_{10}) = 10p(1 - p)$$



$$X_1, X_2 \sim \text{Bern}(p)$$

$$S_2 \sim B(2; p)$$

$$E(S_2) = 2p$$

$$\text{Var}(S_2) = 2p(1 - p)$$

$$X_1, \dots, X_5 \sim \text{Bern}(p)$$

$$S_5 \sim B(5; p)$$

$$E(S_5) = 5p$$

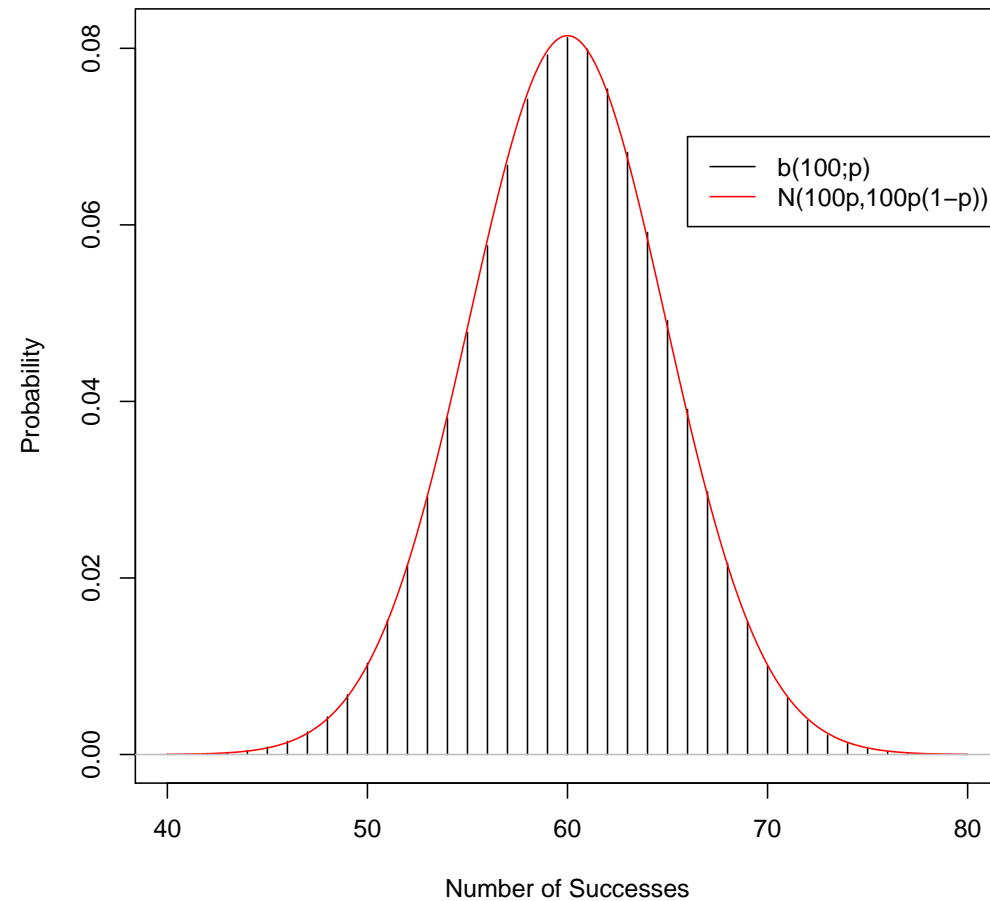
$$\text{Var}(S_5) = 5p(1 - p)$$

$$X_1, \dots, X_{10} \sim \text{Bern}(p)$$

$$S_{10} \sim B(10; p)$$

$$E(S_{10}) = 10p$$

$$\text{Var}(S_{10}) = 10p(1 - p)$$



$$X_1, X_2 \sim \text{Bern}(p)$$

$$S_2 \sim B(2; p)$$

$$E(S_2) = 2p$$

$$\text{Var}(S_2) = 2p(1 - p)$$

$$X_1, \dots, X_5 \sim \text{Bern}(p)$$

$$S_5 \sim B(5; p)$$

$$E(S_5) = 5p$$

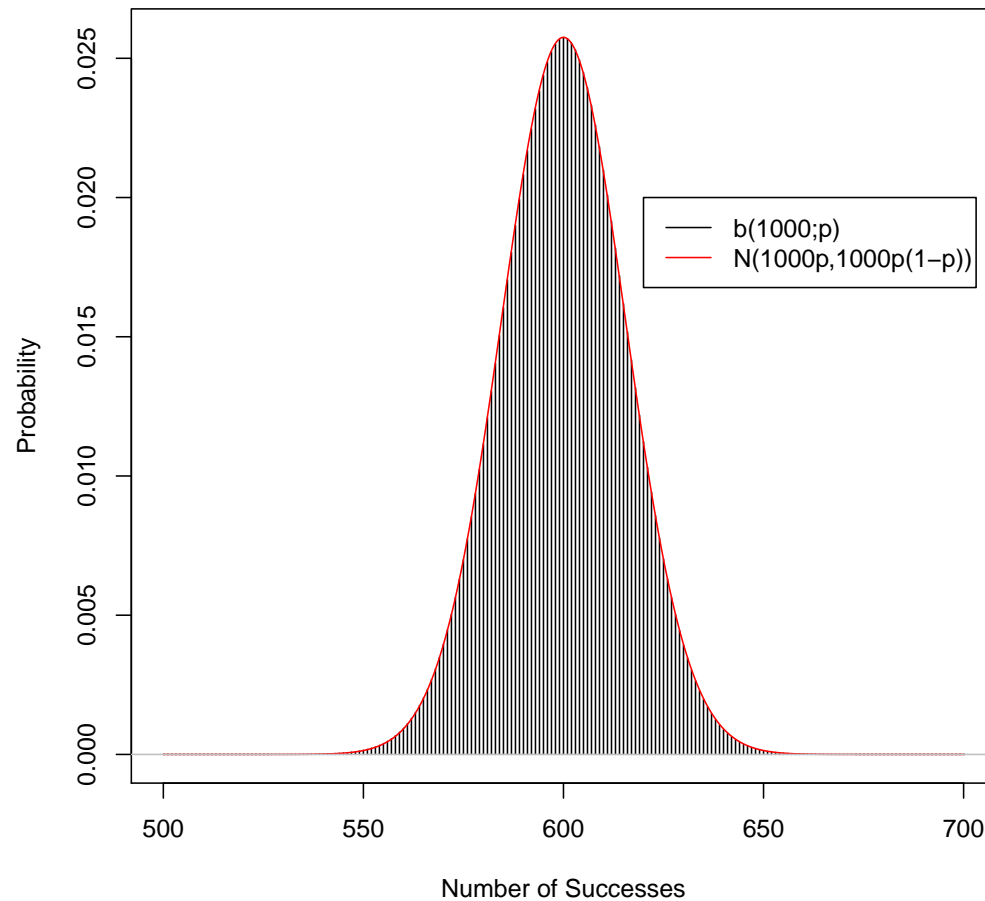
$$\text{Var}(S_5) = 5p(1 - p)$$

$$X_1, \dots, X_{10} \sim \text{Bern}(p)$$

$$S_{10} \sim B(10; p)$$

$$E(S_{10}) = 10p$$

$$\text{Var}(S_{10}) = 10p(1 - p)$$



$$X_1, X_2 \sim \text{Bern}(p)$$

$$S_2 \sim B(2; p)$$

$$E(S_2) = 2p$$

$$\text{Var}(S_2) = 2p(1 - p)$$

$$X_1, \dots, X_5 \sim \text{Bern}(p)$$

$$S_5 \sim B(5; p)$$

$$E(S_5) = 5p$$

$$\text{Var}(S_5) = 5p(1 - p)$$

$$X_1, \dots, X_{10} \sim \text{Bern}(p)$$

$$S_{10} \sim B(10; p)$$

$$E(S_{10}) = 10p$$

$$\text{Var}(S_{10}) = 10p(1 - p)$$

Der zentrale Grenzwertsatz (ZGWS) (1)

In seiner einfachsten Form besagt der ZGWS, dass die Summe einer großen Anzahl von unabhängigen identisch verteilten ZVn approximativ normalverteilt ist.

Erinnerung:

Seien $X_i \sim N(\mu, \sigma^2)$ **unabhängige normalverteilte** ZVn für $i = 1, 2, \dots, n$.

Dann ist die Zufallsvariable $S_n = X_1 + X_2 \dots + X_n$ normalverteilt:
 $S_n \sim N(n\mu, n\sigma^2)$.

Insbesondere gilt für $\overline{X}_n = \frac{S_n}{n}$:

$$\overline{X}_n \sim N\left(\mu, \frac{\sigma^2}{n}\right)$$

Der zentrale Grenzwertsatz (ZGWS) (2)

Satz 4.6

*Der zentrale Grenzwertsatz: Seien X_1, X_2, \dots, X_n **unabhängige identisch** verteilte ZVn mit $E(X_i) = \mu$ und $\text{Var}(X_i) = \sigma^2 > 0$. Weiter sei $S_n = X_1 + X_2 + \dots + X_n$. Dann konvergiert die Verteilung der **Summe** S_n mit wachsendem n gegen die Normalverteilung:*

$$S_n \approx N(n\mu, n\sigma^2), \quad \text{für große } n$$

Insbesondere gilt für $\overline{X}_n = \frac{S_n}{n}$:

$$\overline{X}_n \approx N\left(\mu, \frac{\sigma^2}{n}\right)$$

oder besser gesagt:

$$\overline{X}_n \overset{n \rightarrow \infty}{\rightsquigarrow} N\left(\mu, \frac{\sigma^2}{n}\right) \quad \text{und} \quad S_n \overset{n \rightarrow \infty}{\rightsquigarrow} N(n\mu, n\sigma^2)$$

Der zentrale Grenzwertsatz (ZGWS) (3)

$$\overline{X}_n \overset{n \rightarrow \infty}{\underset{\sim}{\rightsquigarrow}} N\left(\mu, \frac{\sigma^2}{n}\right) \quad \text{und} \quad S_n \overset{n \rightarrow \infty}{\underset{\sim}{\rightsquigarrow}} N(n\mu, n\sigma^2)$$

oder:

$$\frac{\overline{X}_n - \mu}{\frac{\sigma}{\sqrt{n}}} \overset{n \rightarrow \infty}{\underset{\sim}{\rightsquigarrow}} N(0, 1) \quad \text{und} \quad \frac{S_n - n\mu}{\sigma\sqrt{n}} \overset{n \rightarrow \infty}{\underset{\sim}{\rightsquigarrow}} N(0, 1)$$

Der zentrale Grenzwertsatz (ZGWS) (4)

$$\overline{X}_n \overset{n \rightarrow \infty}{\rightsquigarrow} N\left(\mu, \frac{\sigma^2}{n}\right) \quad \text{d.h.} \quad Z_n := \frac{\overline{X}_n - \mu}{\frac{\sigma}{\sqrt{n}}} \overset{n \rightarrow \infty}{\rightsquigarrow} N(0, 1)$$

$$\text{l.a.W.} \quad P\left(\frac{\overline{X}_n - \mu}{\frac{\sigma}{\sqrt{n}}} \leq z\right) \overset{n \rightarrow \infty}{\longrightarrow} \Phi(z)$$

Die Verteilung des standardisierten Mittelwerts (Zufallsvariable!) konvergiert (punktweise) gegen die Standardnormalverteilung.

$$S_n \overset{n \rightarrow \infty}{\rightsquigarrow} N(n\mu, n\sigma^2) \quad \text{oder:} \quad Z_n := \frac{S_n - n\mu}{\sqrt{n}\sigma} \overset{n \rightarrow \infty}{\rightsquigarrow} N(0, 1)$$

$$\text{d.h.} \quad P\left(\frac{S_n - n\mu}{\sqrt{n}\sigma} \leq z\right) \overset{n \rightarrow \infty}{\longrightarrow} \Phi(z)$$

Die Verteilung der standardisierten Summe (Zufallsvariable!) konvergiert (punktweise) gegen die Standardnormalverteilung.

Der zentrale Grenzwertsatz (ZGWS) (4)

$$\overline{X}_n \overset{n \rightarrow \infty}{\rightsquigarrow} N\left(\mu, \frac{\sigma^2}{n}\right) \quad \text{d.h.} \quad Z_n := \frac{\overline{X}_n - \mu}{\frac{\sigma}{\sqrt{n}}} \overset{n \rightarrow \infty}{\rightsquigarrow} N(0, 1)$$

$$\text{l.a.W.} \quad P\left(\frac{\overline{X}_n - \mu}{\frac{\sigma}{\sqrt{n}}} \leq z\right) \xrightarrow{n \rightarrow \infty} \Phi(z)$$

Die Verteilung des standardisierten Mittelwerts (Zufallsvariable!) konvergiert (punktweise) gegen die Standardnormalverteilung.

$$S_n \overset{n \rightarrow \infty}{\rightsquigarrow} N(n\mu, n\sigma^2) \quad \text{oder:} \quad Z_n := \frac{S_n - n\mu}{\sqrt{n}\sigma} \overset{n \rightarrow \infty}{\rightsquigarrow} N(0, 1)$$

$$\text{d.h.} \quad P\left(\frac{S_n - n\mu}{\sqrt{n}\sigma} \leq z\right) \xrightarrow{n \rightarrow \infty} \Phi(z)$$

Die Verteilung der standardisierten Summe (Zufallsvariable!) konvergiert (punktweise) gegen die Standardnormalverteilung.

Der zentrale Grenzwertsatz (ZGWS) mit

$$Z_n := \frac{\overline{X_n} - \mu}{\frac{\sigma}{\sqrt{n}}} \overset{n \rightarrow \infty}{\rightsquigarrow} N(0, 1) \quad \text{oder:} \quad Z_n := \frac{S_n - n\mu}{\sqrt{n}\sigma} \overset{n \rightarrow \infty}{\rightsquigarrow} N(0, 1)$$

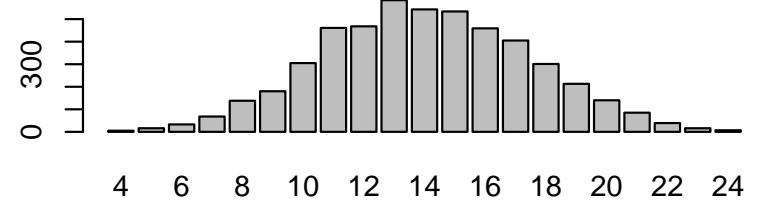
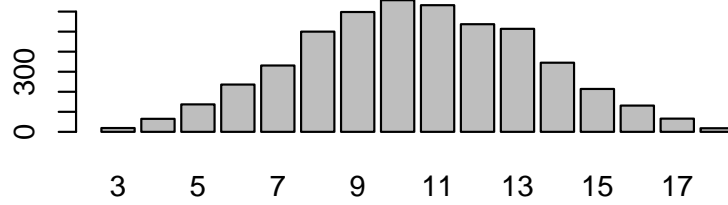
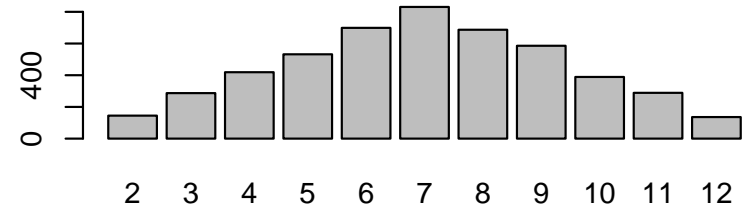
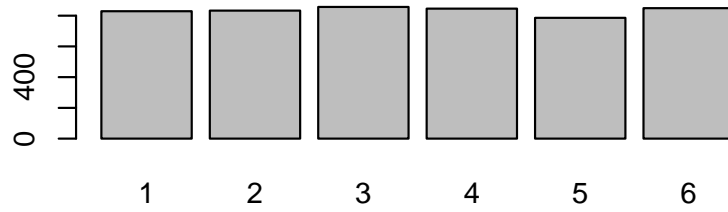
Wie kann man diese Konvergenz der Verteilungen mit  veranschaulichen?

N (für große N) Samples aus dieser Verteilung generieren und damit die Verteilung grafisch darstellen. *Beispiele:*

```
V <- sample(1:6, size=20000, replace=TRUE)
barplot(table(V))
U <- runif(20000, -1, 1)
hist(U, freq=FALSE) # run und siehe die Grafiken!
```


Wir wollen zuerst die Approximation $S_n \approx N(n\mu, n\sigma^2)$ für $n = 1, 2, 3, 4$ veranschaulichen, wenn X_1, X_2, \dots, X_n Zufallszahlen beim n -maligen Würfeln (i.i.d.) sind:

```
N <- 5000; n <- c(1, 2, 3, 4)
op <- par(mfrow = c(2, 2))
for (i in n) {
  W <- sample(1:6, size=N*i, replace=TRUE)
  W <- matrix(W, ncol=i)
  S <- rowSums(W)
  barplot(table(S))
}
par(op)
```



Der zentrale Grenzwertsatz (ZGWS) mit (3)

- 1) Man wählt eine Verteilungsfamilie (binomial, $U[-1, 1]$, ...) und generiert N Samples der Stichprobengröße n (N Mal n Zufallszahlen aus dieser Verteilung).
- 2) Für jeden Datensatz j (1 bis N) berechnet man $s_n^{(j)}$,
- 3) und damit die Werte

$$z_n^{(j)} = \frac{s_n^{(j)} - n\mu}{\sqrt{n}\sigma} \text{ für } j = 1, \dots, N \quad \mu = E(X_i), \sigma = \sqrt{\text{Var}(X_i)}$$

oder

$$z_n^{(j)} = \frac{\overline{x_n^{(j)}} - \mu}{\frac{\sigma}{\sqrt{n}}}.$$

- 4) Man plottet (als Histogramm oder einer geschätzte Dichtefunktion) die Werte $z_n^{(j)}$ (für $1 \leq j \leq N$) und die Dichtefunktion einer $N(0, 1)$ -Verteilung.

Der zentrale Grenzwertsatz (ZGWS) mit (4)

1) Wir generieren eine Zufallsreihe von $n = 200$ Würfelwürfen (x_1, \dots, x_n) und berechnen $s_n = x_1 + x_2 + \dots + x_n$. Dieses Experiment wiederholen wir $N = 2000$ mal und berechnen jeweils die Werte von $s_n^{(1)}, s_n^{(2)}, \dots, s_n^{(N)}$.

2) Wir bilden die Werte der standardisierten Summe:

$$z_n^{(j)} = \frac{s_n^{(j)} - n\mu}{\sqrt{n}\sigma} \text{ für } j = 1, \dots, N \quad \mu = E(X_i), \sigma = \sqrt{\text{Var}(X_i)}$$

3) Wir plotten ein Histogramm für die Werte $z_n^{(j)}$ und fügen eine geschätzte Dichtefunktion zu den Daten in das Histogramm ein.

4) Wir fügen die Dichtefunktion der $N(0, 1)$ -Verteilung dem oben-erstellten Bild hinzu.

1	$x_1^{(1)}$	$x_2^{(1)}$..	$x_n^{(1)}$	$z_n^{(1)}$
2	$x_1^{(2)}$	$x_2^{(2)}$..	$x_n^{(2)}$	$z_n^{(2)}$
⋮	⋮	⋮	⋮	⋮	⋮
j	$x_1^{(j)}$	$x_2^{(j)}$..	$x_n^{(j)}$	$z_n^{(j)}$
⋮	⋮	⋮	⋮	⋮	⋮
N	$x_1^{(N)}$	$x_2^{(N)}$..	$x_n^{(N)}$	$z_n^{(N)}$

Lösung 1

```
N <- 2000; n <- 200 # N Simul. vom Umfang n  
W <- matrix(nrow=N, ncol=n) # Matrix W für N Simul. mit Umfang n  
Zn <- rep(0,N) # Vektor Zn für N standardisierte Summen
```

Lösung 1

```
N <- 2000; n <- 200 # N Simul. vom Umfang n
W <- matrix(nrow=N, ncol=n) # Matrix W für N Simul. mit Umfang n
Zn <- rep(0,N) # Vektor Zn für N standardisierte Summen

for (i in 1:N) {
  # n mal würfeln:
  W[i,] <- sample(1:6, n, replace=TRUE)
```

Lösung 1

```
N <- 2000; n <- 200 # N Simul. vom Umfang n
W <- matrix(nrow=N, ncol=n) # Matrix W für N Simul. mit Umfang n
Zn <- rep(0,N) # Vektor Zn für N standardisierte Summen

for (i in 1:N) {
  # n mal würfeln:
  W[i,] <- sample(1:6, n, replace=TRUE)

  # und berechne die standardisierte Summe:
  Zn[i] <- (1/sqrt(n)) * ((sum(W[i,]) - n*3.5) / sqrt(35/12))
}
# Zn liefert uns die standardisierte Summe aus N Versuchen.
```


Lösung 1

```
N <- 2000; n <- 200 # N Simul. vom Umfang n
W <- matrix(nrow=N, ncol=n) # Matrix W für N Simul. mit Umfang n
Zn <- rep(0,N) # Vektor Zn für N standardisierte Summen

for (i in 1:N) {
  # n mal würfeln:
  W[i,] <- sample(1:6, n, replace=TRUE)

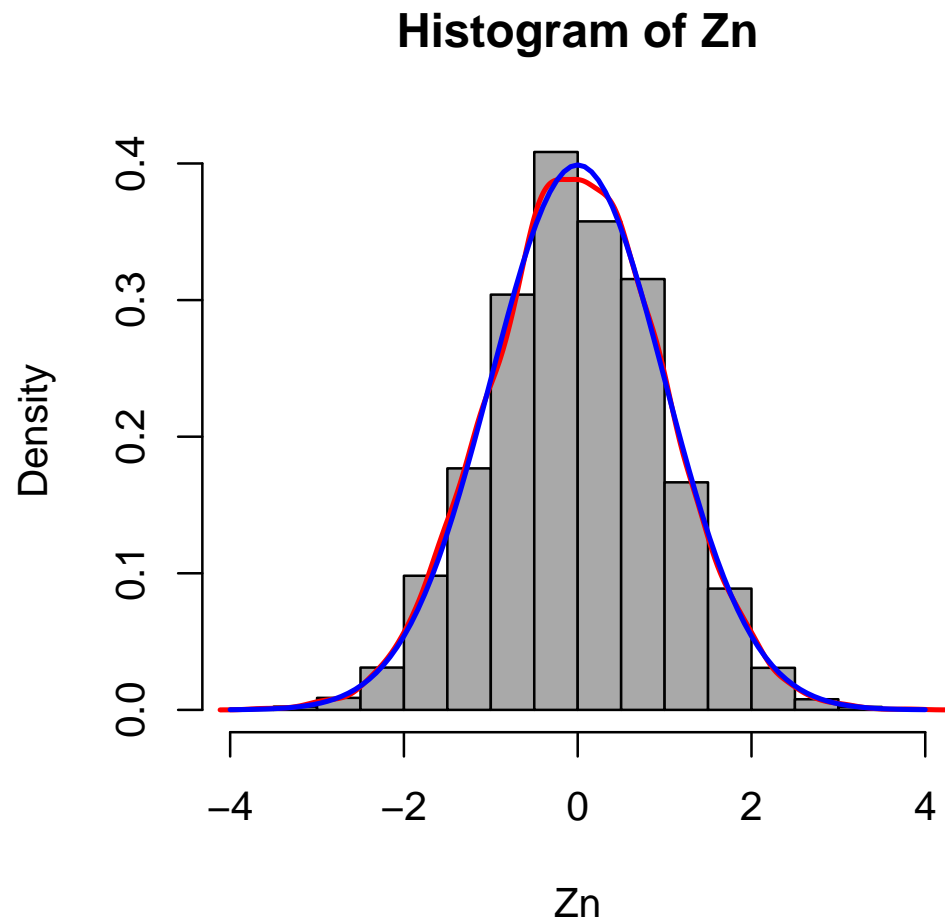
  # und berechne die standardisierte Summe:
  Zn[i] <- (1/sqrt(n)) * ((sum(W[i,]) - n*3.5) / sqrt(35/12))
}
# Zn liefert uns die standardisierte Summe aus N Versuchen.

.....

# oder ohne Schleife:
N <- 2000; n <- 200
W <- sample(1:6, n*N, replace=T) # n*N Randomzahlen
W <- matrix(W, nrow=N, ncol=n) # jedes row(W) enthält eine Sim.
Zn <- (1/sqrt(n)) * ((rowSums(W) - n*3.5) / sqrt(35/12))
```

Lösung 1

```
hist(Zn, col="darkgray", freq=F, xlim=c(-4,4))  
lines(density(x=Zn), col="red", lty=1, lwd=2)  
curve(dnorm(x), add=T, col="blue", lwd=2)
```



Lösung 2

$$Z_n := \frac{S_n - n\mu}{\sqrt{n}\sigma} \overset{n \rightarrow \infty}{\rightsquigarrow} N(0, 1)$$

Wir wollen zeigen, dass man die Verteilung von Z_n mit **wachsendem** n besser durch eine Standardnormalverteilung approximieren kann.

1	$x_1^{(1)}$	$x_2^{(1)}$..	$x_n^{(1)}$	$z_n^{(1)}$
2	$x_1^{(2)}$	$x_2^{(2)}$..	$x_n^{(2)}$	$z_n^{(2)}$
⋮	⋮	⋮	⋮	⋮	⋮
j	$x_1^{(j)}$	$x_2^{(j)}$..	$x_n^{(j)}$	$z_n^{(j)}$
⋮	⋮	⋮	⋮	⋮	⋮
N	$x_1^{(N)}$	$x_2^{(N)}$..	$x_n^{(N)}$	$z_n^{(N)}$

$$n = 1 \Rightarrow z_n^{(j)} = z_1^{(j)} = \frac{s_1^{(j)} - 1\mu}{\sqrt{1}\sigma} = \frac{x_1^{(j)} - 1\mu}{\sqrt{1}\sigma}; j = 1, \dots, N$$

$$n = 2 \Rightarrow z_n^{(j)} = z_2^{(j)} = \frac{s_2^{(j)} - 2\mu}{\sqrt{2}\sigma} = \frac{(x_1^{(j)} + x_2^{(j)}) - 2\mu}{\sqrt{2}\sigma}; j = 1, \dots, N$$

$$n = n \Rightarrow z_n^{(j)} = \frac{s_n^{(j)} - n\mu}{\sqrt{n}\sigma} = \frac{(x_1^{(j)} + \dots + x_n^{(j)}) - n\mu}{\sqrt{n}\sigma}; j = 1, \dots, N$$

1	$x_1^{(1)}$	$x_2^{(1)}$..	$x_n^{(1)}$	$z_n^{(1)}$
2	$x_1^{(2)}$	$x_2^{(2)}$..	$x_n^{(2)}$	$z_n^{(2)}$
⋮	⋮	⋮	⋮	⋮	⋮
j	$x_1^{(j)}$	$x_2^{(j)}$..	$x_n^{(j)}$	$z_n^{(j)}$
⋮	⋮	⋮	⋮	⋮	⋮
N	$x_1^{(N)}$	$x_2^{(N)}$..	$x_n^{(N)}$	$z_n^{(N)}$

$$n = 1 \Rightarrow z_n^{(j)} = z_1^{(j)} = \frac{s_1^{(j)} - 1\mu}{\sqrt{1}\sigma} = \frac{x_1^{(j)} - 1\mu}{\sqrt{1}\sigma}; j = 1, \dots, N$$

$$n = 2 \Rightarrow z_n^{(j)} = z_2^{(j)} = \frac{s_2^{(j)} - 2\mu}{\sqrt{2}\sigma} = \frac{(x_1^{(j)} + x_2^{(j)}) - 2\mu}{\sqrt{2}\sigma}; j = 1, \dots, N$$

$$n = n \Rightarrow z_n^{(j)} = \frac{s_n^{(j)} - n\mu}{\sqrt{n}\sigma} = \frac{(x_1^{(j)} + \dots + x_n^{(j)}) - n\mu}{\sqrt{n}\sigma}; j = 1, \dots, N$$

1	$x_1^{(1)}$	$x_2^{(1)}$..	$x_n^{(1)}$	$z_n^{(1)}$
2	$x_1^{(2)}$	$x_2^{(2)}$..	$x_n^{(2)}$	$z_n^{(2)}$
⋮	⋮	⋮	⋮	⋮	⋮
j	$x_1^{(j)}$	$x_2^{(j)}$..	$x_n^{(j)}$	$z_n^{(j)}$
⋮	⋮	⋮	⋮	⋮	⋮
N	$x_1^{(N)}$	$x_2^{(N)}$..	$x_n^{(N)}$	$z_n^{(N)}$

$$n = 1 \Rightarrow z_n^{(j)} = z_1^{(j)} = \frac{s_1^{(j)} - 1\mu}{\sqrt{1}\sigma} = \frac{x_1^{(j)} - 1\mu}{\sqrt{1}\sigma}; j = 1, \dots, N$$

$$n = 2 \Rightarrow z_n^{(j)} = z_2^{(j)} = \frac{s_2^{(j)} - 2\mu}{\sqrt{2}\sigma} = \frac{(x_1^{(j)} + x_2^{(j)}) - 2\mu}{\sqrt{2}\sigma}; j = 1, \dots, N$$

$$n = n \Rightarrow z_n^{(j)} = \frac{s_n^{(j)} - n\mu}{\sqrt{n}\sigma} = \frac{(x_1^{(j)} + \dots + x_n^{(j)}) - n\mu}{\sqrt{n}\sigma}; j = 1, \dots, N$$

Lösung 2: $n \rightsquigarrow \infty$

```
N <- 10000; n <- c(seq(1, 4, 1), seq(100, 1000, 100))
```

Lösung 2: $n \rightsquigarrow \infty$

```
N <- 10000; n <- c(seq(1,4,1), seq(100,1000,100))
```

```
graphics.off()  
par(mfrow = c(2,2), pty = "s")
```


Lösung 2: $n \rightsquigarrow \infty$

```
N <- 10000; n <- c(seq(1,4,1), seq(100,1000,100))

graphics.off()
par(mfrow = c(2,2), pty = "s")

for (i in n) {
  W <- sample(1:6, i*N, replace=T)
  W <- matrix(W, ncol=i) # Matrix W für N Simul. mit Umfang i
  # und berechne ihre standardisierte Summe:
  Zn <- (1/sqrt(i)) * ((rowSums(W)-i*3.5)/sqrt(35/12))
}
```

Lösung 2: $n \rightsquigarrow \infty$

```
N <- 10000; n <- c(seq(1,4,1), seq(100,1000,100))

graphics.off()
par(mfrow = c(2,2), pty = "s")

for (i in n) {
  W <- sample(1:6, i*N, replace=T)
  W <- matrix(W, ncol=i) # Matrix W für N Simul. mit Umfang i
  # und berechne ihre standardisierte Summe:
  Zn <- (1/sqrt(i)) * ((rowSums(W)-i*3.5)/sqrt(35/12))

  plot(density(x=Zn), col="red", lwd=2,
        main=paste("n=",i), xlim=c(-4,4), ylim=c(0,0.7))
  curve(dnorm(x), add=T, col="blue", lwd=2)
  Sys.sleep(1)
}
```

Lösung 3: $n \rightsquigarrow \infty$ mit *qqnorm()*

```
N <- 10000; n <- c(seq(1, 4, 1), seq(100, 1000, 100))
```

Lösung 3: $n \rightsquigarrow \infty$ mit *qqnorm()*

```
N <- 10000; n <- c(seq(1, 4, 1), seq(100, 1000, 100))  
  
graphics.off()  
par(mfrow = c(2, 2), pty = "s")
```

Lösung 3: $n \rightsquigarrow \infty$ mit *qqnorm()*

```
N <- 10000; n <- c(seq(1,4,1), seq(100,1000,100))

graphics.off()
par(mfrow = c(2,2), pty = "s")

for (i in n) {
  W <- sample(1:6, i*N, replace=T)
  W <- matrix(W, ncol=i) # Matrix W für N Simul. mit Umfang i
  # und berechne ihre standardisierte Summe:
  Zn <- (1/sqrt(i)) * ((rowSums(W)-i*3.5)/sqrt(35/12))
}
```

Lösung 3: $n \rightsquigarrow \infty$ mit *qqnorm()*

```
N <- 10000; n <- c(seq(1,4,1), seq(100,1000,100))

graphics.off()
par(mfrow = c(2,2), pty = "s")

for (i in n) {
  W <- sample(1:6, i*N, replace=T)
  W <- matrix(W, ncol=i) # Matrix W für N Simul. mit Umfang i
  # und berechne ihre standardisierte Summe:
  Zn <- (1/sqrt(i)) * ((rowSums(W)-i*3.5)/sqrt(35/12))

  qqnorm(Zn, main=paste("n=",i), xlim=c(-3,3), ylim=c(-3,3))
  abline(0, 1, col=2)
  Sys.sleep(1)
}
```

Übung 4.5

Aus dem ZGWS folgt:

$$\frac{\frac{1}{n} \sum_{i=1}^n X_i - \mu}{\sigma / \sqrt{n}} \approx N(0, 1)$$

Mit Hilfe von Simulationsstudien veranschaulichen Sie diese Folgerung für verschiedene Verteilungen.

Gesetze der großen Zahlen mit

Mit wachsendem Stichprobenumfang konvergiert der Mittelwert der Stichprobe gegen den Erwartungswert:

$$\hat{X}_n = \frac{X_1 + X_2 + \dots + X_n}{n} \longrightarrow E(X_i) \quad \text{stochastisch / P-f.s.}$$

Übung 4.6

Es sei $\Omega := \{1, 2, 3, 4, 5, 6\}$ der Ereignisraum bei einmaligem Würfeln. Weiter sei X_i auf Ω definiert

$$X_i = \begin{cases} 1 & \omega = 6 \\ 0 & \text{sonst} \end{cases}$$

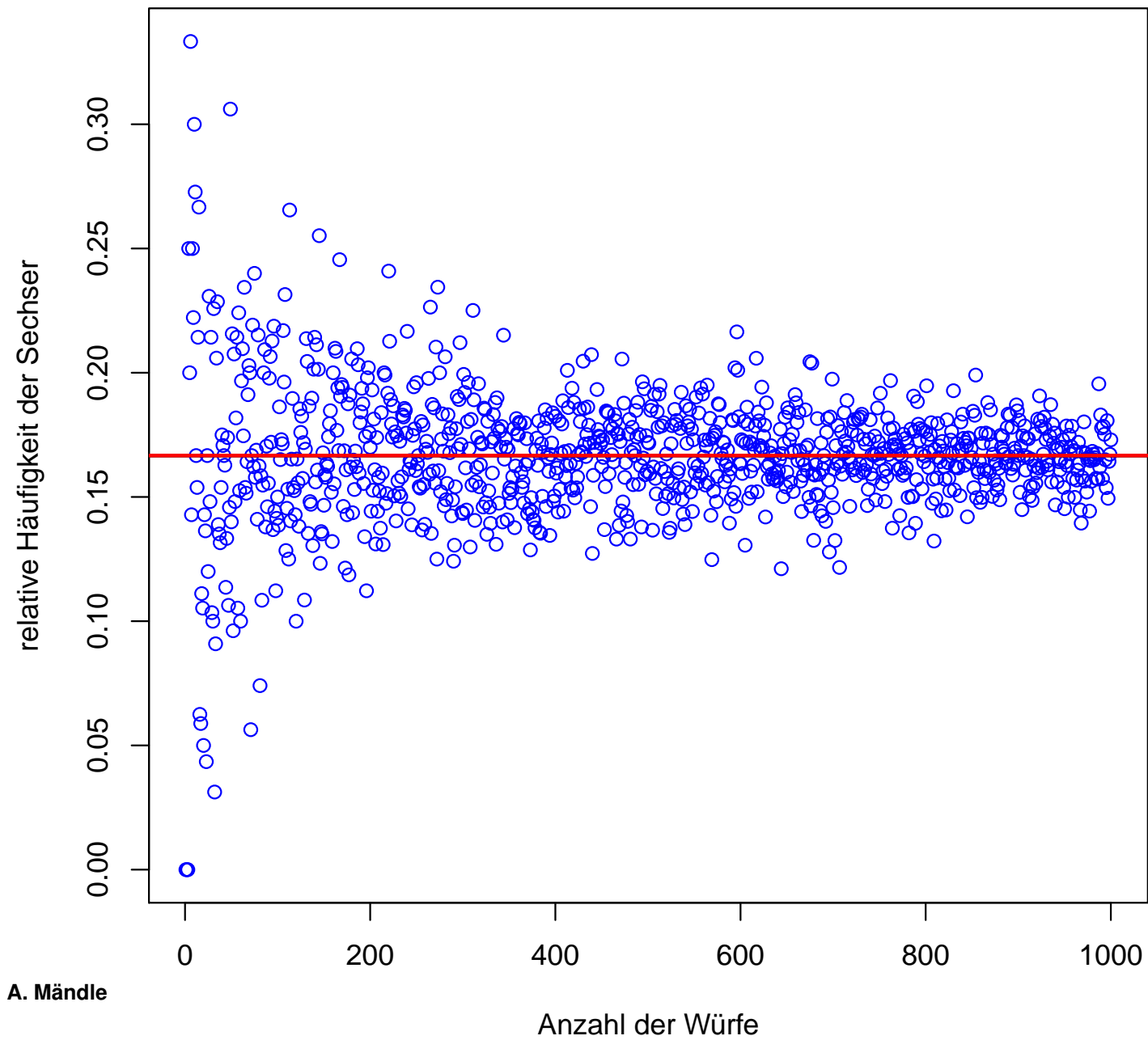
wobei $i = 1, 2, \dots, n$ ist. S_j gibt die Anzahl der Sechser bei j -maligem Würfeln an

$$S_j = \sum_{i=1}^j X_i \quad \text{für } j = 1, 2, \dots, n$$

und $\hat{X}_j = S_j/j$, die relative Häufigkeit der Sechser bei j -maligem Würfeln, liefert uns den Mittelwert von X_1, X_2, \dots, X_j . Untersuchen Sie die Konvergenz von $\{\hat{X}_j\}_{j=1,2,\dots,n}$ gegen $E(X_i)$ mittels Visualisierung der Stabilisierung der relativen Häufigkeiten der *Sechser*.

Lösung

```
my.fun <- function(n) {  
  x.hat <- rep(0,n)  
  for (i in 1:n) {  
    w <- sample(1:6, size=i, replace=T)  
    x.hat[i] <- sum(w==6)/i  
  }  
  x.hat  
}  
t <- my.fun(n=1000)  
plot(1:length(t), t, xlab="Anzahl_der_Würfe",  
      ylab="relative_Häufigkeit_der_Sechser",  
      lwd=1, col="blue")  
abline(1/6, 0, col="red", lwd=2)
```



$$\int_a^b g(x) dx = ?$$

1. Generiere n (unabhängige) ZV X_1, X_2, \dots, X_n aus $U[a, b]$.

2. $\frac{1}{n} \sum_{i=1}^n g(X_i) \longrightarrow E(g(X)).$

3. $E(g(X)) = \int_a^b g(x) f_{X_i}(x) dx = \int_a^b g(x) \frac{1}{b-a} dx.$

4. Folglich $(b-a) \frac{1}{n} \sum_{i=1}^n g(X_i) \longrightarrow \int_a^b g(x) dx$

Übung: Monte Carlo-Integration

Übung 4.7

a. Berechnen Sie per Monte Carlo-Simulation und stellen Sie die Funktionsgraphen in den gegebenen Bereichen dar.

$$\int_{-1}^1 e^{x^2} dx$$

$$\int_0^{\pi/8} \log(1 + \tan^2(x)) dx$$

$$\int_{-\pi/2}^{\pi/2} \sin(2x) \cos(x) dx$$

Übung: Monte Carlo-Integration

Übung 4.7

a. Berechnen Sie per Monte Carlo-Simulation und stellen Sie die Funktionsgraphen in den gegebenen Bereichen dar.

$$\int_{-1}^1 e^{x^2} dx$$

$$\int_0^{\pi/8} \log(1 + \tan^2(x)) dx$$

$$\int_{-\pi/2}^{\pi/2} \sin(2x) \cos(x) dx$$

```
2*mean(exp(runif(1000000,min=-1,max=1)^2))
```

Übung: Monte Carlo-Integration

b. Stellen Sie das Konvergenzverhalten Ihrer Simulationsmethode in Teil a. graphisch dar.

(Hinweis: schätzen Sie das Integral für $n = 1, 2, \dots, 10000$ (n = Anzahl der Simulationen) und stellen Sie die Integralschätzungen für $n = 1, 2, \dots, 10000$ in einer Abbildung dar.)

Die Funktion `integrate()` berechnet ein Integral numerisch. Verwenden Sie diese Funktion, um die o.g. Integrale zu berechnen und visualisieren Sie das Konvergenzverhalten der berechneten Werte aus der M.C. Simulation gemeinsam mit dem numerisch berechneten Wert aus der Funktion `integrate()`.

Übung: Monte-Carlo-Simulation

Übung 4.8

Wir wollen die Zahl π näherungsweise per M.C. Simulation berechnen.

Erzeugen Sie dazu B Zufallszahlen Z in einem Quadrat $[-1, 1] \times [-1, 1]$. Dann zählen Sie, wie viele Punkte im Kreis $K(0, 1)$ liegen. Das Verhältnis der Anzahl der Punkte im Kreis zu der Anzahl der Punkte insgesamt entspricht näherungsweise dem Verhältnis von der Fläche des Kreises zu der Fläche des Rechtecks. Dadurch lässt sich die Zahl π approximieren. Schreiben Sie eine Funktion

```
my.pi <- function(B=1000) {  
  x.B <- runif(B, min=?, max=?)  
  y.B <- ...  
  ...  
}
```

die dies umsetzt.

Monte Carlo-Simulation

Anwendungen:

- Berechnung eines Integrals
- Verteilung einer Zufallsvariable (Teststatistik) berechnen
- Power eines statistischen Tests berechnen

Gliederung I

Organisatorisches

1. Einstieg und Grundlegendes zu 	42
2. Datentypen und Datenimport	92
3. Deskriptive Statistik mit <i>R</i>	193
4. Verteilungen & Zufallszahlen in 	349
5. Schliessende Statistik: Testen und Schätzen	441
5.1 Binomialtest	446
5.2 t-Test	502
5.3 Multiples Testen	549
5.4 F-Test	572
5.5 Anpassungstests	582

Gliederung II

5.6 Testen auf Unabhängigkeit	603
5.7 Varianzanalyse	618
5.8 Varianzanalyse und Regressionsanalyse	647
5.9 Regressionsanalyse	648

Test-Hypothesen: H_0 und H_1

Wahrer Zustand (in der Grundgesamtheit)	Entscheidung (aufgrund des Stichprobenergebnisses)	
H_0 ist richtig	H_0 beibehalten Spezifität $1 - \alpha$	H_0 verwerfen \Rightarrow Fehler erster Art α
H_1 ist richtig	H_0 beibehalten \Rightarrow Fehler zweiter Art β	H_0 verwerfen Power $1 - \beta$

Fehler 1. Art (α - Fehler): Wahrscheinlichkeit, mit der die Nullhypothese fälschlicherweise abgelehnt wird.

Fehler 2. Art (β - Fehler): Wahrscheinlichkeit, mit der die Alternativhypothese fälschlicherweise nicht als richtig erkannt wird.

Power des Tests: Wahrscheinlichkeit, mit der die Alternativhypothese richtig erkannt wird.



α - und β - Fehler

Erläuterung durch bedingte Wahrscheinlichkeiten:

α	P(Entscheidung für H_1 H_0 ist richtig)
$1 - \alpha$	P(Entscheidung für H_0 H_0 ist richtig)
β	P(Entscheidung für H_0 H_1 ist richtig)
$1 - \beta$	P(Entscheidung für H_1 H_1 ist richtig)

$1 - \alpha$ = Spezifität eines Tests

$1 - \beta$ = Sensitivität/Power/Güte/Stärke oder Trennschärfe

Parametrische Tests	bzgl. (für k Stichpr.)	in 
Binomialtest	Anteilswert	<code>binom.test</code>
t-Test	Lage vom Mittelwert, $k \in \{1, 2\}$	<code>t.test</code>
Varianzanalyse	Gleichheit der Mittelwerte ($k > 2$)	<code>oneway.test</code>
Korrelationstest	Unabhängigkeit	<code>cor.test</code>
Shapiro-Wilk-test	Normalverteilung	<code>shapiro.test</code>
F-Test	Varianzhomogenität ($k = 2$)	<code>var.test</code>
Bartlett-Test	Varianzhomogenität ($k > 2$)	<code>bartlett.test</code>
Nicht-Parametrische Tests:	bzgl. (für k Stichpr.)	in 
Wilcoxon-Test	Lage vom Mittelwert, $k \in \{1, 2\}$	<code>wilcox.test</code>
Kruskal-Wallis-Test	wie Wilcoxon, aber $k > 2$	<code>kruskal.test</code>
Kolmogorov-Smirnov-Test	Anpassungsgüte	<code>ks.test</code>
χ^2 -Anpassungstest	Anpassungsgüte	<code>chisq.test</code>
χ^2 -Unabhängigkeitstest	Unabhängigkeit	<code>chisq.test</code>
...

Statistische Tests

- 1 Binomialtest
- 2 t-Test (Mittelwertvergleich)
 - 2.1 Der Einstichproben t-Test
 - 2.2 Der Zweistichproben t-Test (für gepaarte und ungepaarte Stichproben)
- 3 Multiple Test
- 4 F-Test (Varianzvergleich)
- 5 Anpassungstests
- 6 Testen auf Unabhängigkeit
 - 6.1 χ^2 -Test auf Unabhängigkeit
 - 6.2 Korrelationstest auf Unabhängigkeit
- 7 Varianzanalyse (Mittelwertvergleich $n > 2$)
- 8 Regressionsanalyse

Binomialtest – Testhypothesen

<i>Einseitiger unterer Binomialtest</i>	$H_0 : p \geq p_0$	$H_1 : p < p_0$
<i>Einseitiger oberer Binomialtest</i>	$H_0 : p \leq p_0$	$H_1 : p > p_0$
<i>Zweiseitiger Binomialtest</i>	$H_0 : p = p_0$	$H_1 : p \neq p_0$

Beispiel: *einseitiger oberer Binomialtest*

Medikament A wirkt gegen eine bestimmte Krankheit mit **bekannter** Heilungswahrscheinlichkeit $p_0 = P(A) = 0.3$

Neues Medikament B wirkt mit **unbekannter** Wk. $p = P(B)$

$$H_0: p \leq p_0 \text{ (oder } H_0: p = p_0) \quad H_1: p > p_0$$

Studienergebnis zu Medikament B: 15 von 40 Patienten geheilt.

$$\Rightarrow \hat{p} = \frac{15}{40} = 0.375$$

Um die Skeptiker zu überzeugen, müssen wir die Nullhypothese entkräften; d.h. zeigen, dass unter der Nullhypothese die Beobachtung sehr unwahrscheinlich ist.

Anzahl der Erfolge X ist binomialverteilt:

$$X \sim B(40; p)$$

Wir haben beobachtet $X = 15$.

Wäre das Heilungsverhältnis 40 von 40 oder 39 von 40, hätten wir uns intuitiv für H_1 entschieden, weil unter H_0 ($p \leq 0.3$) diese Ereignisse sehr unwahrscheinlich scheinen.

Ab welcher Heilungsquote, k von 40, sollte man H_0 ablehnen?

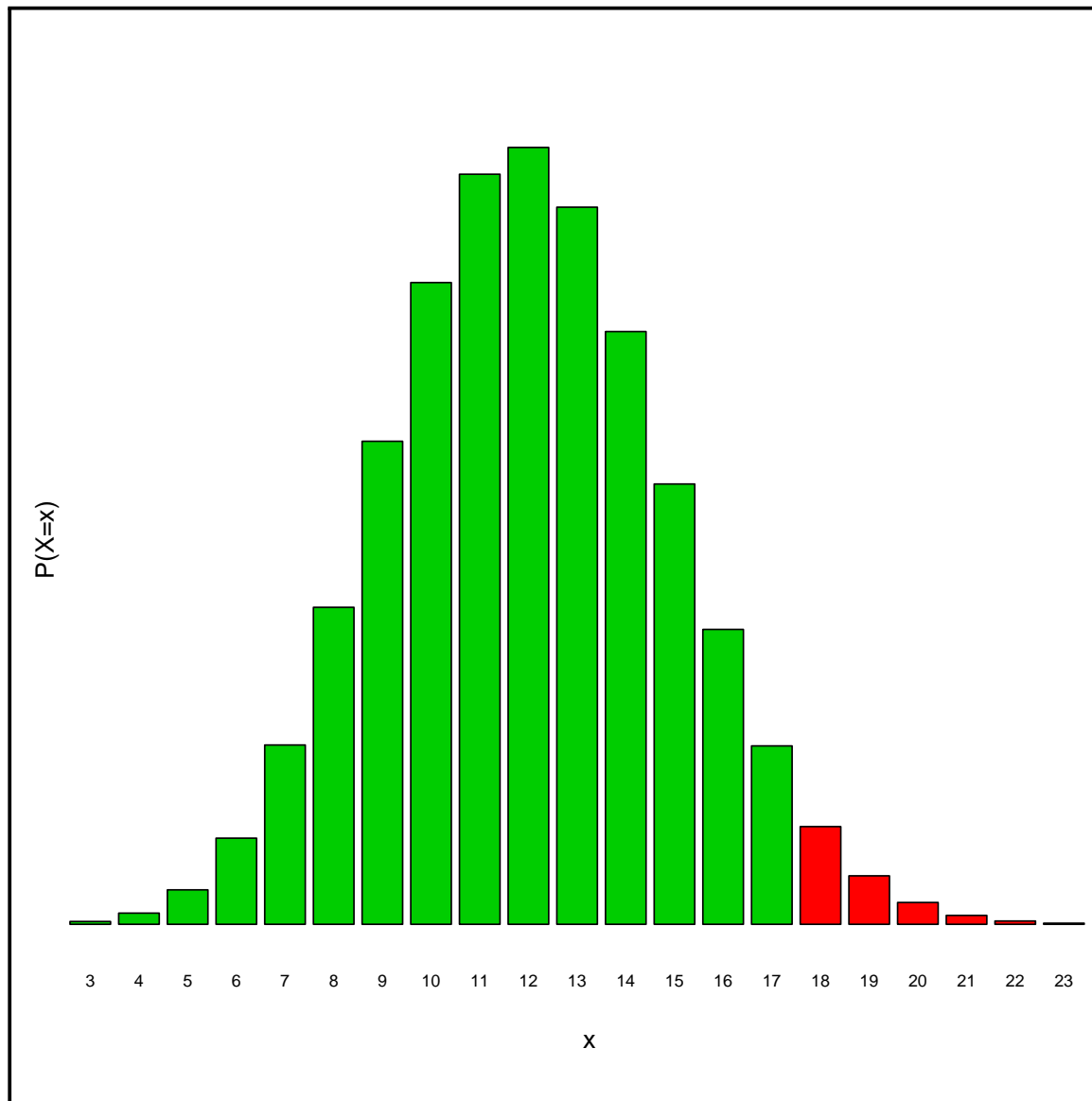
Suche kleinstes k , sodass (Ergebnis unter H_0 unwahrscheinlich, d.h.)

$$P(X > k \mid H_0) = P(X > k \mid p \leq 0.3) \leq \alpha$$

für ein vorgegebenes (kleines) Signifikanzniveau α . Wir setzen $\alpha = 0.05$ und $p = 0.3$:

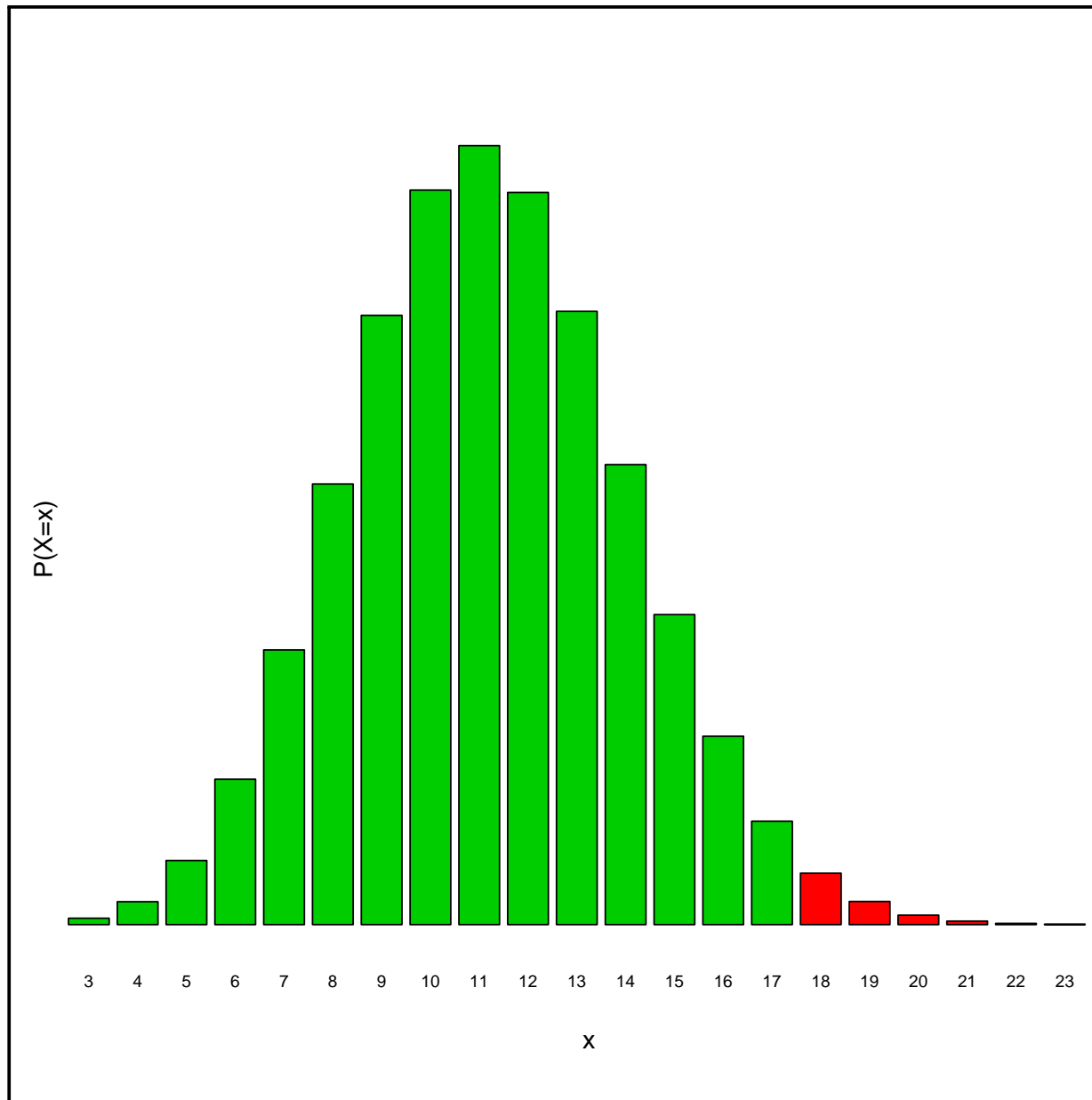
k	$P(X > k \mid p = 0.3)$	k	$P(X > k \mid p = 0.3)$
15	0.115	17	0.032
16	0.063	18	0.015

Also ist die Menge $\{X > 17\}$ für $p = 0.3$ *kritisch* (unwahrscheinlich).

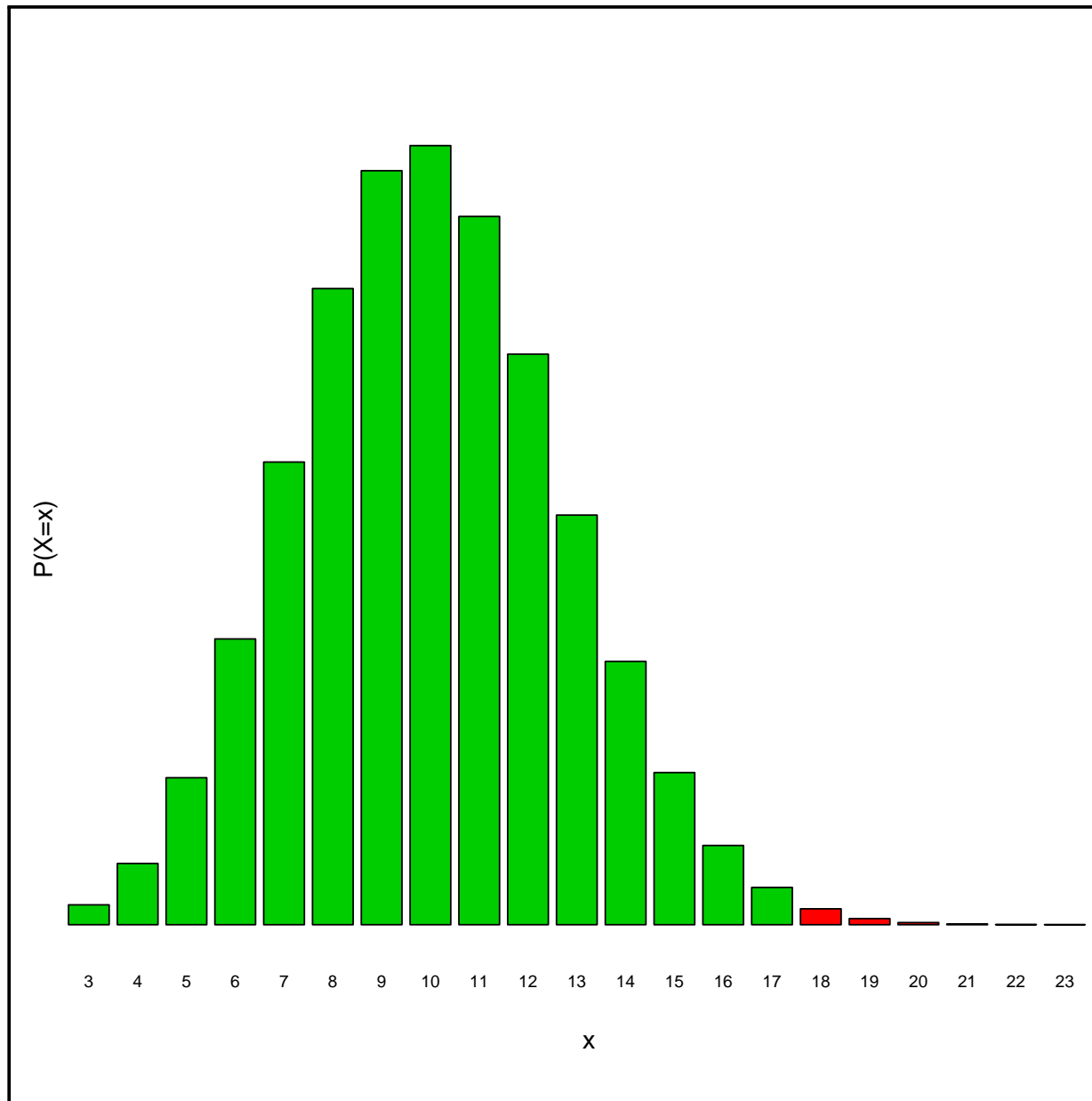


$$P(X > 17 \mid p = 0.3) = 0.03 \leq 0.05$$

Wenn $\{X > 17\}$ für $p = 0.3$ kritisch ist, dann auch für $p < 0.3$:



z.B. $p = 0.28$: $P(X > 17 \mid p = 0.28) = 0.02 \leq 0.05$



oder $p = 0.25$: $P(X > 17 \mid p = 0.25) = 0.005 \leq 0.05$

Wir haben festgestellt:

$$P(X > 17 \mid p = 0.3) = 0.032 \leq 0.05$$

$$P(X > 16 \mid p = 0.3) = 0.063 > 0.05$$

\Rightarrow für $p \leq 0.3$ sind die Werte $X > 17$ unwahrscheinlich!

$c = 17$: kritischer Wert für $X \sim b(40, 0.3)$

$B_{H_1} = \{18, 19, \dots, 40\}$: kritischer (Ablehnungs-) Bereich

$B_{H_0} = \{1, 2, \dots, 17\}$: Nicht-Ablehnungsbereich

In unserer Studie war $X = 15$, hat also den kritischen Wert nicht überschritten. Daher kann man die Nullhypothese H_0 nicht verwerfen (mit α -Fehler 0.05).

Heilungserfolge in 15 von 40 Fällen reichen nicht aus, um daraus zu schließen, dass das neue Medikament besser als das Alte ist.

- ▶ Annahme: H_0 ist richtig. Mögliche Werte für X ?
- ▶ Theoretisch: $X = 0, 1, 2, \dots, 40$
- ▶ aber: manche Werte sind „unwahrscheinlich“, z.B. $X = 40, 39$. Welche Werte?
- ▶ Man legt α fest, z.B. 0.05 (Fehler 1. Art : die W-keit, mit der die Nullhypothese fälschlicherweise abgelehnt wird.)
- ▶ Man definiert „unwahrscheinliche“ Werte, als den kritischen Bereich B_{H_1} , d.h. diejenigen X , die zusammen höchstens α ausmachen und am stärksten für H_1 sprechen.
- ▶ $B_{H_1} = \textit{kritischer Bereich} = \{18, 19, \dots, 40\}$

Testentscheidung:

- ▶ 1. Falls $X \in B_{H_1}$: H_0 wird zugunsten von H_1 verworfen.
- ▶ 2. Falls $X \in B_{H_0}$: H_0 wird nicht abgelehnt.

Ein statistischer Nachweis gelingt bloß dann, wenn die Nullhypothese verworfen wird. Belassen der Nullhypothese bedeutet dagegen nicht, dass H_0 damit statistisch bewiesen ist!

Wie berechnen wir den Fehler 2. Art

Fehler 2. Art (β -Fehler): die Wahrscheinlichkeit an, mit der die Alternativhypothese fälschlicherweise nicht als richtig erkannt wird.

Die Wahrscheinlichkeit eines Fehlers 2. Art kann man nur für ein konkretes p , welches zur Alternative gehört, berechnen; z.B. $p = 0.4$. Es sei $Y \sim B(40, 0.4)$:

Der Fehler wird begangen, wenn $Y \in B_{H_0}$ oder $Y \leq 17$ ist:

$$\beta = P(Y \leq 17 \mid p = 0.4) = 0.69$$

Interpretation: Obwohl $p = 0.4$ *deutlich* größer als $p_0 = 0.3$ ist, wird diese Tatsache mit einer W-keit von 0.69 nicht entdeckt.

Power des Tests: $1 - \beta = 0.31$

Wie kann man den Fehler 2. Art verringern?

Bei festem α kann man den Fehler 2. Art durch eine größere Stichprobenzahl n verringern.

Wäre im Medikamentenvergleich-Beispiel $n = 200$, dann findet man den kritischen Wert c durch:

$$P(X > c \mid p = 0.3) \leq 0.05 \quad \text{und} \quad P(X > c - 1 \mid p = 0.3) > 0.05$$

$$P(X > 71 \mid p = 0.3) = 0.04 \leq 0.05 \quad \text{und} \quad P(X > 70 \mid p = 0.3) = 0.054 > 0.05$$

$$\Rightarrow B_{H_1} = \{72, 73, \dots, 200\}, \quad B_{H_0} = \{0, 1, \dots, 71\}$$

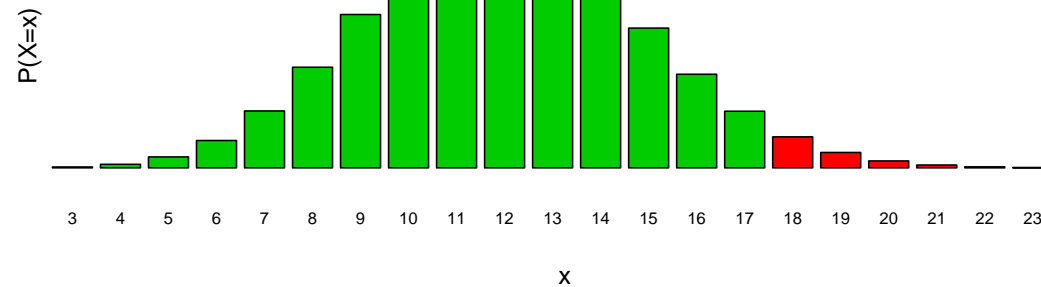
Fehler 2. Art für $Y \sim B(200, p = 0.4)$:

$$\beta = P(Y \leq 71 \mid p = 0.4) = 0.11$$

Power des Tests: $1 - \beta = 0.89$

Schlussfolgerung

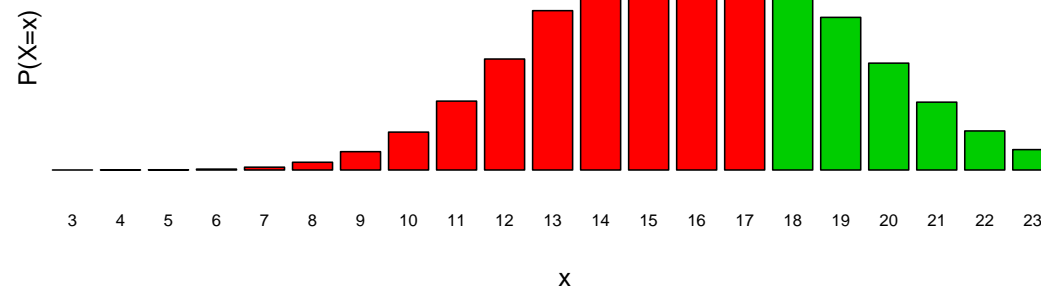
- ▶ Bei der Studie mit 40 Patienten reicht die Erfolgsquote von 37.5% (15 von 40) nicht aus um den überlegenen therapeutischen Nutzen des neuen Medikamentes mit 40% Erfolgsquote *statistisch nachzuweisen*.
- ▶ Hätten wir die Studie mit 200 Patienten durchgeführt, hätte eine Erfolgsquote von 36% (72 von 200) ausgereicht, um den therapeutischen Nutzen des neuen Medikamentes mit 40% Erfolgsquote *statistisch nachzuweisen*.



α -Fehler = 0.05

$H_0 : X \sim B(40; 0.3)$

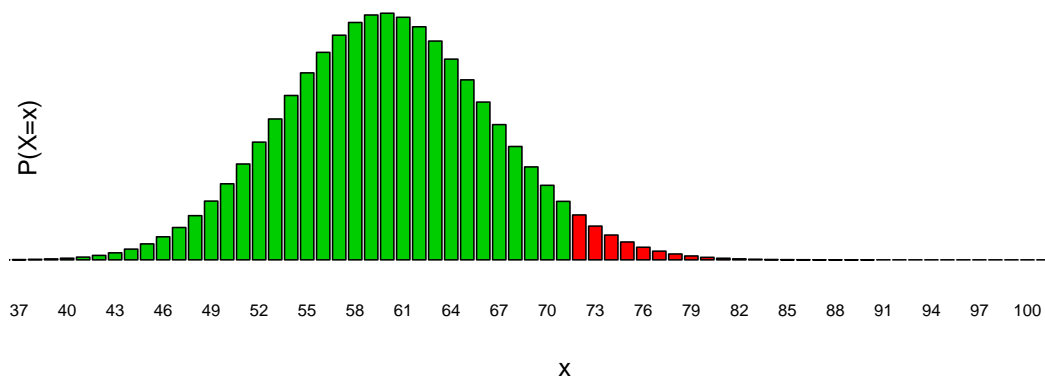
$P(X > 17) \approx 0.05$



berechne β -Fehler:

$H_1 : Y \sim B(40; 0.4)$

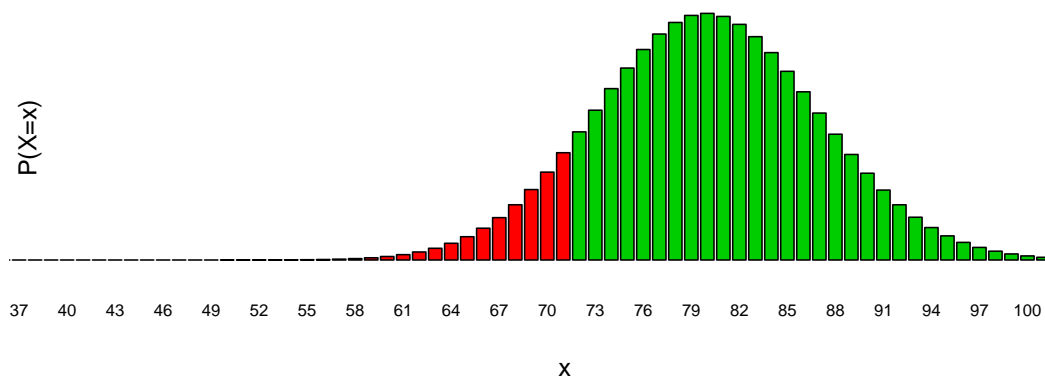
$P(Y \leq 17) \approx 0.69$



α -Fehler = 0.05

$H_0 : X \sim B(200; 0.3)$

$P(X > 71) \approx 0.05$



berechne β -Fehler:

$H_1 : Y \sim B(200; 0.4)$

$P(Y \leq 71) \approx 0.11$

Einseitiger *oberer* Binomialtest

- ▶ $H_0 : p \leq p_0$ vs. $H_1 : p > p_0$
- ▶ Bezeichne c den kritischen Wert für $X \sim B(n, p_0)$:
$$P(X > c) \leq \alpha \quad \text{und}$$
$$P(X > c - 1) = P(X \geq c) > \alpha$$
- ▶ Kritischer Bereich: $B_{H_1} = \{X > c\} = \{c + 1, c + 2, \dots\}$
- ▶ Falls der beobachtete Wert von X in B_{H_1} liegt:
 H_0 zugunsten von H_1 verwerfen.
- ▶ Falls der beobachtete Wert von X nicht in B_{H_1} liegt:
 H_0 beibehalten.

Einseitiger *unterer* Binomialtest

- ▶ $H_0 : p \geq p_0$ vs. $H_1 : p < p_0$
- ▶ Bezeichne c den kritischen Wert für $X \sim B(n, p_0)$:
$$P(X < c) \leq \alpha \quad \text{und}$$
$$P(X < c + 1) = P(X \leq c) > \alpha$$
- ▶ Kritischer Bereich: $B_{H_1} = \{X < c\} = \{0, 1, \dots, c - 1\}$
- ▶ Falls der beobachtete Wert von X in B_{H_1} liegt:
 H_0 zugunsten von H_1 verwerfen.
- ▶ Falls der beobachtete Wert von X nicht in B_{H_1} liegt:
 H_0 beibehalten.

Zweiseitiger Binomialtest

- ▶ $H_0 : p = p_0$ vs. $H_1 : p \neq p_0$
- ▶ Bezeichne c_1 und c_2 die kritischen Werte für $X \sim B(n, p_0)$:

$$P(X < c_1) \leq \alpha/2 \quad \text{und}$$

$$P(X < c_1 + 1) = P(X \leq c_1) > \alpha/2$$

$$P(X > c_2) \leq \alpha/2 \quad \text{und}$$

$$P(X > c_2 - 1) = P(X \geq c_2) > \alpha/2$$

- ▶ Kritischer Bereich: $B_{H_1} = \{X < c_1\} \cup \{X > c_2\}$
- ▶ $B_{H_1} = \{0, 1, \dots, c_1 - 1\} \cup \{c_2 + 1, c_2 + 2, \dots, n\}$
- ▶ Falls der beobachtete Wert von X in B_{H_1} liegt:
 H_0 zugunsten von H_1 verwerfen.
- ▶ Falls der beobachtete Wert von X nicht in B_{H_1} liegt:
 H_0 beibehalten.

?binom.test

Exakter Test einer einfachen Nullhypothese über die Erfolgswahrscheinlichkeit in einem Bernoulli-Experiment.

```
binom.test(x, n, p = 0.5,  
alternative = c("two.sided", "less", "greater"),  
conf.level = 0.95)
```

Argumente

x Anzahl der Erfolge, oder Vektor der Länge 2 aus Anzahl der Erfolge und Anzahl der Misserfolge.

n Anzahl der Versuche; ignoriert, falls x Länge 2 hat.

p unterstellte Erfolgswahrscheinlichkeit.

alternative Alternativhypothese: entweder "two.sided", "less" oder "greater". (Anfangsbuchstabe reicht).

conf.level Konfidenzniveau für Konfidenzintervalle.

?binom.test

Werte

statistic *Anzahl der Erfolge.*

parameter *Anzahl der Versuche.*

p.value *p-Wert des Tests.*

conf.int *Konfidenz-Intervall für die Erfolgswahrscheinlichkeit.*

estimate *geschätzte Erfolgswahrscheinlichkeit.*

null.value *Erfolgswahrscheinlichkeit unter der Nullhypothese.*

alternative *Zeichenkette, Beschreibung der Alternativhypothese.*

method *Zeichenkette "Exact binomial test".*

data.name *Zeichenkette, Namen der Daten.*

Beispiel: Zweiseitiger Binomialtest

In einer Stadt wurden in einem Jahr 13452 Jungen und 12860 Mädchen geboren. Testen Sie die Nullhypothese H_0 , dass die Wahrscheinlichkeit für eine Knabengeburt 0.5 beträgt, zum Niveau 0.05.

```
binom.test(x=c(13452,12860), p=0.5 , conf.level=0.95)
```

Exact binomial test

data: c(13452, 12860)

number of successes = 13452, number of trials = 26312,

p-value = 0.0002689

alternative hypothesis: true probability of success is not equal to 0.5

95 percent confidence interval:

0.5051897 0.5173070

sample estimates:

probability of success

0.5112496

Beispiel: Einseitiger *unterer* Binomialtest

Vorherige Studien haben gezeigt, dass 4 Prozent aller Fische in einem Teich eine bestimmte Krankheit haben. Man behauptet, dass nach Durchführung bestimmter Maßnahmen, der Anteil der Kranken gesunken ist. In einer neuen Studie war von 100 Fischen nur einer krank. Kann man der Behauptung mit einer Irrtumswahrscheinlichkeit unter 0.02 vertrauen?

```
binom.test(x=1,n=100,p=.04,alternative="l",conf.level=.98)
```

Exact binomial test

data: 1 and 100

number of successes = 1, number of trials = 100, p-value = 0.08716

alternative hypothesis: true probability of success is less than 0.04

98 percent confidence interval:

0.00 0.0569

sample estimates:

probability of success

0.01

Übung 5.1

Ein Spieler wurde disqualifiziert, nachdem er 10 mal nacheinander beim 10 maligen Würfelspiel die Sechs gewürfelt hatte. War der Vorwurf *berechtigt*? Begründen Sie Ihre Antwort mit dem Binomialtest.

Lösung

```
p <- 1/6 ; n <- k <- 10 ; alpha <- 0.01  
binom.test(k, n, p, "g")
```

Exact binomial test

data: k and n

number of successes = 10, number of trials = 10, p-value = 1.654e-08

alternative hypothesis: true probability of success is greater than 0.167

99 percent confidence interval:

0.63 1

sample estimates:

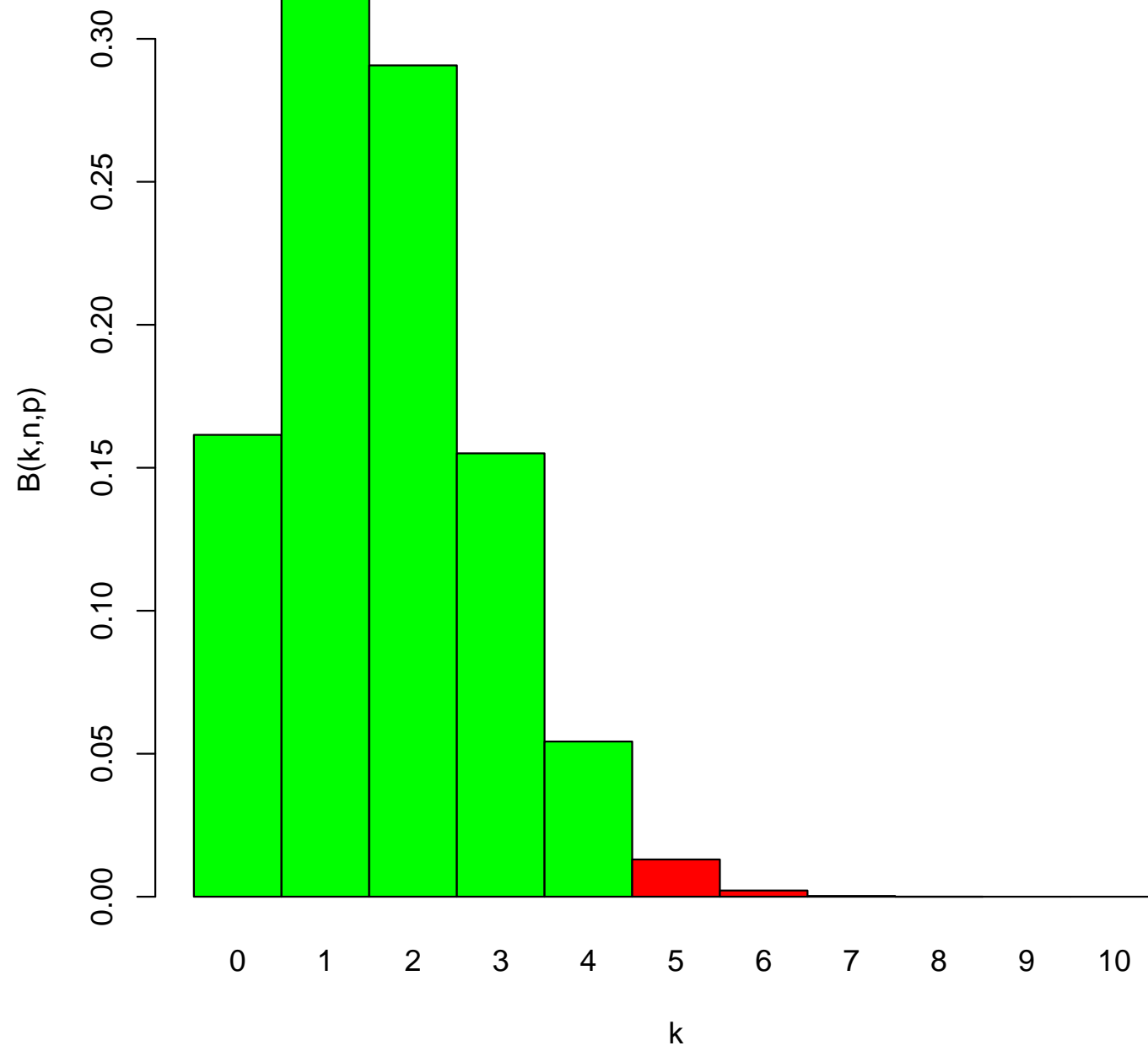
probability of success

1

```

p <- 1/6 ; n <- 10 ; alpha <- 0.01
x <- 0:10
dichte <- dbinom(x, n, p) # P(X = k)
vert <- pbinom(x, n, p) # P(X <= k)
col1 <- rep("green", sum(vert < (1-alpha)))
col2 <- rep("red", sum(vert >= (1-alpha)))
col <- c(col1, col2)
barplot(names=x, height=dichte, col=col, space=0,
        xlab="k", ylab="B(k,n,p)")

```



Test-Entscheidung

$p\text{-Wert} \leq \alpha \Rightarrow H_0$ verwerfen, Entscheidung für H_1

$p\text{-Wert} > \alpha \Rightarrow H_0$ beibehalten

p -Wert eines Tests ...

- deutet an, wie glaubhaft es ist, die Beobachtungen zu erhalten, wenn die Nullhypothese wahr ist.
- ist das kleinste Signifikanzniveau, zu dem die Beobachtungen die Annahme der Alternative rechtfertigen.
- ist das kleinste Signifikanzniveau α , bei dem die Nullhypothese verworfen wird.

Übung 5.2

Eine Pharmafirma hat ein neues Medikament entwickelt, von dem vermutet wird, dass es die Heilungschance bei einer bestimmten Krankheit von 30% (Erfolgschance bei Standardmedikation) auf etwa 40% erhöht. Man führt eine Studie mit $n = 200$ Patienten durch.

Wie groß ist die Chance, den vermuteten Effekt (ca. 10%) mittels der Studie nachweisen zu können? Verwenden Sie das Signifikanzniveau $\alpha = 0.05$.

Hinweis: Chance den vermuteten Effekt nachzuweisen = Wahrscheinlichkeit einer Entscheidung für H_1 , wenn H_1 richtig ist (Power des Tests), d.h.

$$P(\text{entscheiden für } H_1 \mid H_1 \text{ ist richtig}) = 1 - \beta$$

Lösung

Einseitiger oberer Binomialtest: $H_0 : p \leq 0.3$ vs. $H_1 : p > 0.3$

$$X \sim B(200; 0.3) \quad Y \sim B(200; 0.4)$$

Gesucht: die Wahrscheinlichkeit sich für H_1 zu entscheiden, wenn $Y \sim B(200; 0.4)$:

$$P(Y \in B_{H_1}) = P(Y > c) = 1 - P(Y \leq c)$$

Zuerst finden wir c durch:

$$P(X > c) \leq 0.05 \quad \text{oder} \quad P(X \leq c) \geq 0.95$$

```
qbinom(p=0.95, size=200, prob=0.3) # oder  
qbinom(p=0.05, size=200, prob=0.3, F)  
71 # also  $k = 71$ 
```

```
1-pbinom(q=71, size=200, prob=0.4) # oder  
pbinom(q=71, size=200, prob=0.4, F)  
0.8906 # also Chance für erfolgr. Nachweis  $\approx 0.89$ 
```

- ▶ $H_0 : p = p_0$ vs. $H_1 : p = p_1$ mit $p_1 > p_0$
- ▶ gegeben: α, n, p_0 und $\delta = p_1 - p_0$
- ▶ Berechne c , den kritischen Wert für $X \sim B(n, p_0)$:

$$P(X > c) \leq \alpha \quad \text{und}$$

$$P(X \geq c) > \alpha$$

- ▶ Berechne Power für $Y \sim B(n, p_1)$:
Power = $P(Y > c) = 1 - P(Y \leq c)$

```
c <- qbinom(p=1-alpha, size=n, prob=p0)
power <- 1-pbinom(q=c, size=n, prob=p1)
```

- $H_0 : p = p_0$ vs. $H_1 : p = p_1$ mit $p_1 < p_0$
- gegeben: α, n, p_0 und $\delta = p_0 - p_1$
- Berechne c , den kritischen Wert für $X \sim B(n, p_0)$:

$$P(X < c) \leq \alpha \quad \text{und}$$

$$P(X \leq c) > \alpha$$

- Berechne Power für $Y \sim B(n, p_1)$:
Power = $P(Y < c) = P(Y \leq c - 1)$

```
c <- qbinom(p=alpha, size=n, prob=p0)
power <- pbinom(q=c-1, size=n, prob=p1)
```

- $H_0 : p = p_0$ vs. $H_1 : p = p_1$ mit $|p_1 - p_0| = \delta$
- gegeben: α, n, p_0 und $\delta = |p_1 - p_0|$
- Berechne c_1 und c_2 , die kritischen Werte für $X \sim B(n, p_0)$:
$$P(X < c_1) \leq \alpha/2 \quad \text{und} \quad P(X \leq c_1) > \alpha/2$$
$$P(X > c_2) \leq \alpha/2 \quad \text{und} \quad P(X \geq c_2) > \alpha/2$$
- Berechne Power mit $Y_1 \sim B(n, p_{1,1})$ und $Y_2 \sim B(n, p_{1,2})$ mit
 $p_{1,1} = p_0 - \delta$ und $p_{1,2} = p_0 + \delta$:
$$\text{Power.1} = P(Y_1 < c_1) = P(Y_1 \leq c_1 - 1)$$
$$\text{Power.2} = P(Y_2 > c_2) = 1 - P(Y_2 \leq c_2)$$

Power des 2-seitigen Binomialtests berechnen (2)

Power = min(Power.1, Power.2)

```
c1 <- qbinom(p=alpha/2, size=n, prob=p0)
```

```
c2 <- qbinom(p=1-alpha/2, size=n, prob=p0)
```

```
power.1 <- pbinom(q=c1-1, size=n, prob=p0 - delta)
```

```
power.2 <- 1-pbinom(q=c2, size=n, prob=p0 + delta)
```

```
power <- min(Power.1, Power.2)
```

Power des oberen Binomialtests $power(n)$

$Power(\alpha, n, p_0, \delta)$:

```
k <- qbinom(p=1- $\alpha$ , size=n, prob= $p_0$ )  
power <- 1-pbinom(q=k, size=n, prob= $p_0 + \delta$ )
```

Setzt man α , p_0 und δ fest, dann erhält man:

$Power(n)$:

```
k <- qbinom(p=1- $\alpha$ , size=n, prob= $p_0$ )  
power <- 1-pbinom(q=k, size=n, prob= $p_0 + \delta$ )
```

Wir schreiben eine Funktion, um $Power(n)$ für versch. Werte von n auszuwerten (für feste α , p_0 und δ).

Power des oberen Binomialtests $power(n)$

`power(n): n=size`

```
k <- qbinom(p=1-0.05, size, prob=0.3)
power <- 1-pbinom(q=k, size, prob=0.3+0.1)
```

```
powerfkt1 <- function(alpha=0.05, p0=0.3, delta=0.1,
                      nstart=25, nend=1000, nstep=25){
  n <- seq(nstart, nend, nstep)
  power <- NULL; alpha.act <- NULL
  for (i in 1:length(n)) {
    cv <- qbinom(p=1-alpha, size=n[i], prob=p0)
    alpha.act[i] <- 1-pbinom(cv, n[i], p0)
    power[i] <- 1-pbinom(q=cv, size=n[i], prob=p0+delta)
  }
  x <- data.frame(n, power, alpha.act)
  return(x)
}

b <- powerfkt1()
```


Power	n	alpha.act
0.27	25	0.044
0.44	50	0.047
0.54	75	0.041
0.62	100	0.034
0.74	125	0.041
.	.	
0.95	250	0.044
1.00	500	0.045
.	.	

Bemerkung: beim Binomialtest wird das vorgegebene α nicht stets voll ausgeschöpft. \Rightarrow Powerverlust!

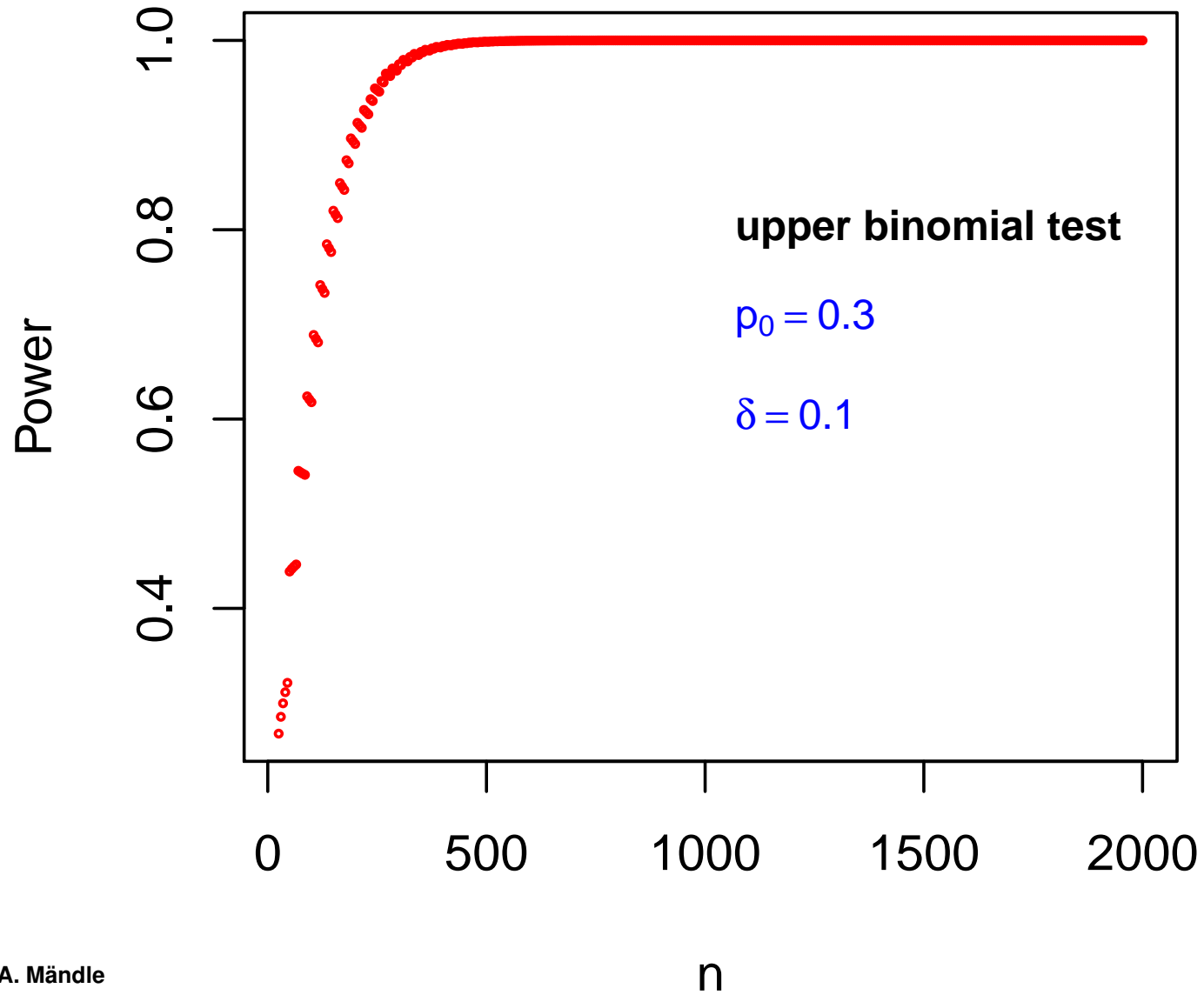
```

# Plot: Power vs. Sample size
b <- powerfkt1(nstep=5, nend=2000)
plot(b$n, b$power, xlab="n", ylab="Power", col="red",
      main="Power_vs._sample_size", cex=0.5)
text(1000, .8, "upper_binomial_test",
      cex=.8, pos=4, font=2)
text(1000, .7, expression(p[0]==.3),
      cex=.8, pos=4, font=1, col="blue")
text(1000, .6, expression(delta==.1),
      cex=.8, pos=4, font=1, col="blue")
abline(0,0)

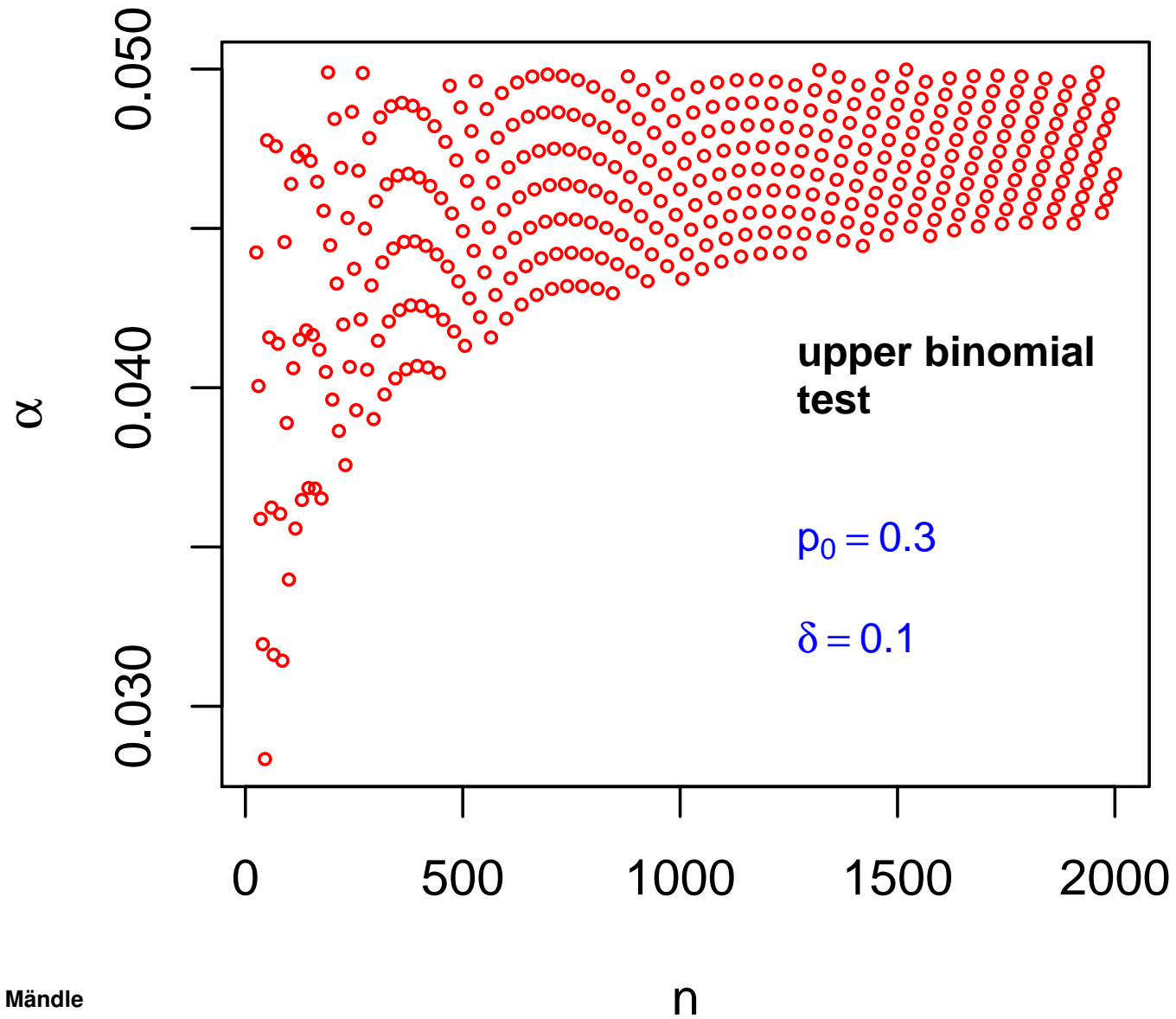
# Plot: alpha vs. Sample size
plot(b$n, b$alpha.act, xlab="n", ylab=expression(alpha),
      col="red", main=paste(expression(alpha),
        "vs._sample_size", sep="_"), cex=0.5)
text(1200, .04, "upper_binomial_test",
      cex=.8, pos=4, font=2)
text(1200, .035, expression(p[0]==.3),
      cex=.8, pos=4, font=1, col="blue")
text(1200, .032, expression(delta==.1),
      cex=.8, pos=4, font=1, col="blue")
abline(0,0)

```

Power vs. sample size



alpha vs. sample size



Übung 5.3

Die Power eines oberen Binomialtests $1 - \beta$ hängt von α , n , p_0 und δ ab.

Setzen Sie feste Werte für α , n und p_0 in die Funktion `powerfkt1` ein und berechnen Sie die Power für verschiedene δ .

Power des oberen-Binomial-Tests $power(\delta)$

$power(\delta)$

```
k <- qbinom(p=1-0.05, size=200, prob=0.3)
power <- 1-pbinom(q=k, size=200, prob=0.3+0.1)
```

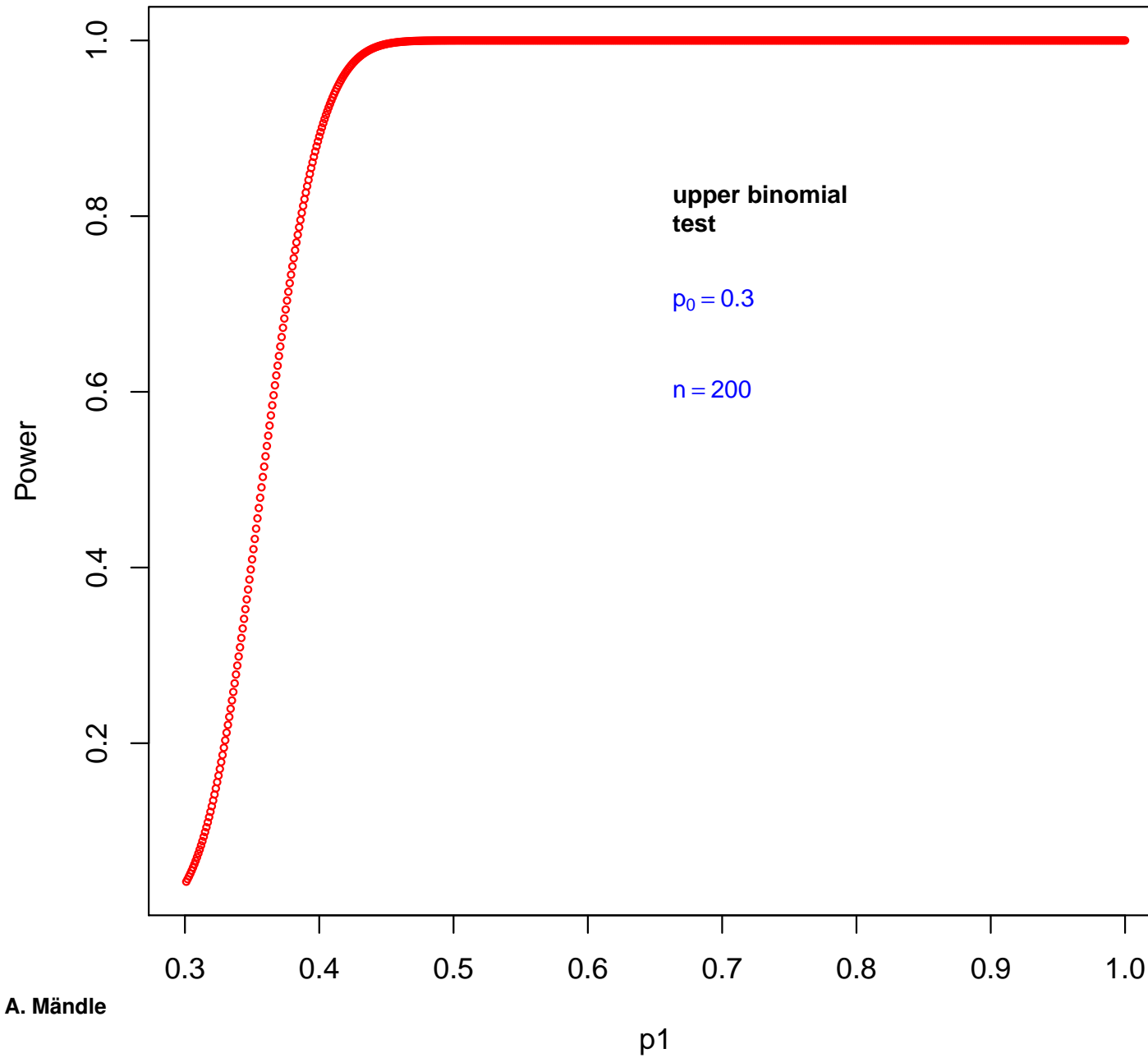
```
powerfkt2 <- function(alpha=0.05, p0=0.3, size1=200,
                      p.end=1, dstep=0.01) {
  p1 <- seq(p0+dstep, p.end, dstep)
  cv <- qbinom(p=1-alpha, size=size1, prob=p0)
  alpha.act <- rep(1-pbinom(cv, size=size1, p0), length(p1))
  power <- NULL
  for (i in 1:length(p1)) {
    power[i] <- 1-pbinom(q=cv, size=size1, prob=p1[i])
  }
  x <- data.frame(p1, power, alpha.act)
  return(x)
}

b <- powerfkt2(dstep=0.001)
```

p_1	Power	alpha.act
0.301	0.042	0.039
0.302	0.045	0.039
.	.	.
0.320	0.128	0.039
.	.	.
0.370	0.640	0.039
.	.	.
0.400	0.890	0.039
.	.	.
0.439	0.990	0.039
.	.	.

```
# Plot: Power vs. p1
b <- powerfkt2(dstep=0.001)
plot(b$p1, b$power, xlab="p1", ylab="Power", col="red",
      main="Power_vs._p1", cex=0.5)
text(.65, .8, "upper_binomial_test",
      cex=.8, pos=4, font=2)
text(.65, .7, expression(p[0]==.3),
      cex=.8, pos=4, font=1, col="blue")
text(.65, .6, expression(n==200),
      cex=.8, pos=4, font=1, col="blue")
abline(0,0)
```


Power vs. p1



Übung 5.4

Schreiben Sie eine Funktion, welche die Power des oberen Binomialtests für fixierte Werte von α und p_0 und verschiedene Werte von n und δ liefert und stellen Sie das Ergebnis grafisch dar.

Power des oberen-Binomial-Tests $power(n, \delta)$

```
powerfkt3 <- function(alpha=0.05, p0=0.3,
nstart=25, nend=1000, nstep=25, p.end=1, dstep=0.01){
  p1 <- seq(p0+dstep, p.end, dstep)
  n1 <- length(p1)
  n <- seq(nstart, nend, nstep)
  n2 <- length(n)
  power <- matrix(rep(0,n1*n2), ncol=n1)
  for(i in 1:n2){
    power[i,] <- powerfkt2(alpha=alpha, p0=p0, size1=n[i],
                           p.end=p.end, dstep=dstep)$power
  }
  x <- t(rbind(p1, round(power, 3)))
  colnames(x) <- c("P1",
                  paste("n=", seq(n[1], n[length(n)], nstep), sep=""))
  return(list("Power"=x, "N"=n))
}

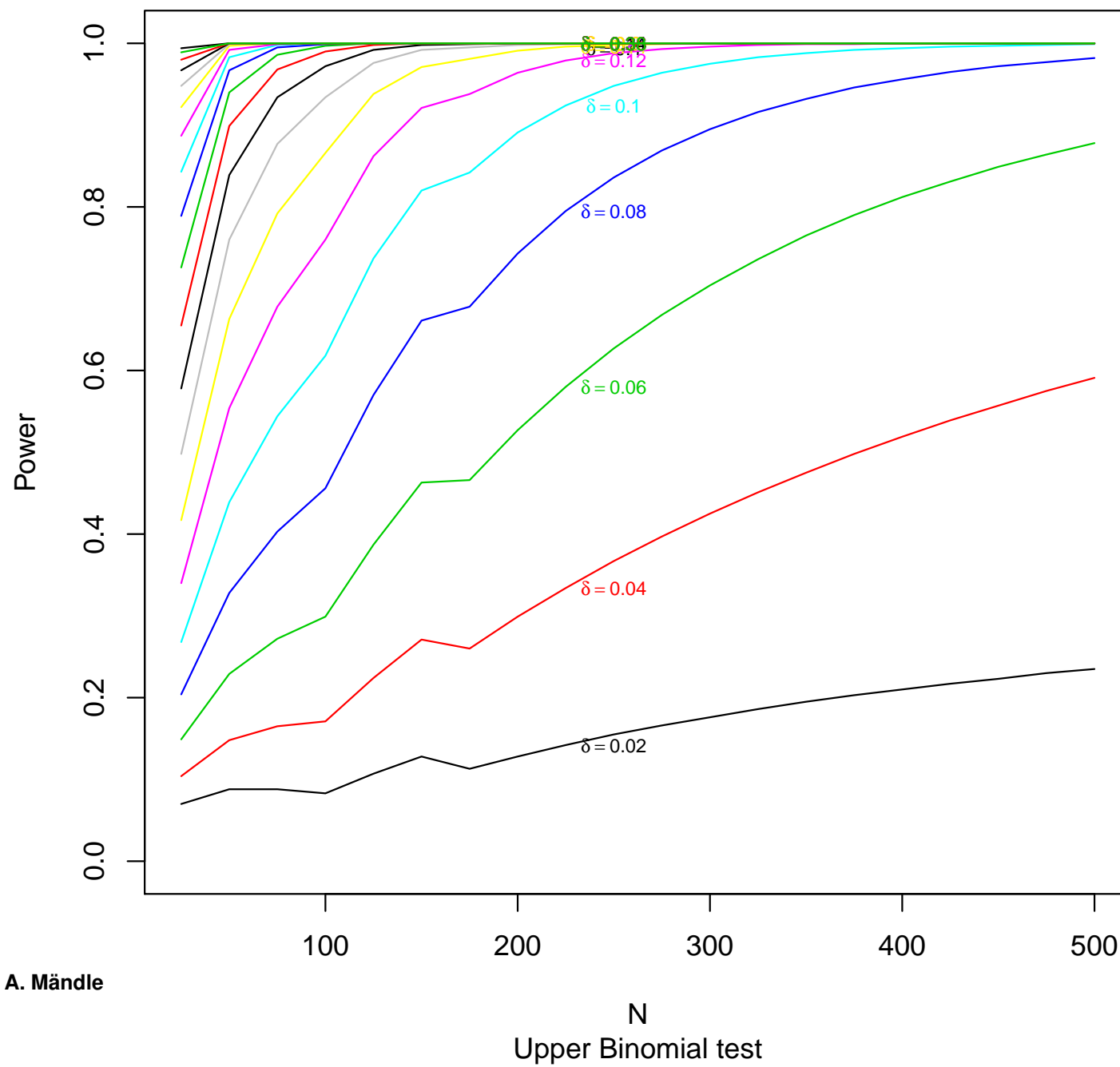
b <- powerfkt3(dstep=0.01, nend=500)
```

```

# Plot: Power(delta, n)
b <- powerfkt3(dstep=0.02, nend=500, p.end=0.7)
p0 <- 0.3 ; x <- b$N ; y <- b$Power
n2 <- nrow(y) ; n1 <- length(x)
plot(x, y[n2,-1], type="l", ylim=c(0,1),
      xlab="N", ylab="Power",
      main=expression(paste(Power(delta,n),
                             "_for_:", alpha==0.05, "_and_", p[0]==0.3)),
      sub="Upper_Binomial_test")
text(x[10], y[n2,2], bquote(delta ==.(y[n2,1]-p0)),
      cex=0.7, col=1)
for(i in 1:(n2-1)){
  lines(x, y[i,-1], col=i)
  text(x[10], y[i,10], bquote(delta ==.(y[i,1]-p0)),
        cex=0.7, col=i)
}
abline(0,0)

```

Power(δ , n) for : $\alpha = 0.05$ and $p_0 = 0.3$



package: binom

binom.power(): Benutzt Wald-Statistik zum Berechnen von Power-Kurven.

binom.confint(): Benutzt acht verschiedene Methoden um Konfidenzintervalle für die Binomialwahrscheinlichkeiten zu bestimmen.

Definition: Das zweiseitige Konfidenzintervall

Definition 5.1

Es seien x_1, x_2, \dots, x_n Beobachtungen einer Stichprobe.

Angenommen die Beobachtungen sind eine Realisation der unabhängigen und identisch-verteilten (i.i.d.) ZV-n X_1, X_2, \dots, X_n mit unbekanntem Verteilungs-Parameter θ (z.B. der Parameter p einer Binomialverteilung). Ein **zweiseitiges Konfidenzintervall** für den Parameter θ **zum Niveau $1 - \alpha$** (bzw. mit Irrtumswahrscheinlichkeit α) ist

ein Intervall: $[\hat{\theta}_u, \hat{\theta}_o]$

(u: unten, o: oben), das den *wahren* Parameter θ mit (*mindestens*) der Wahrscheinlichkeit $1 - \alpha$ überdeckt:

$$P\left(\theta \in [\hat{\theta}_u, \hat{\theta}_o]\right) = 1 - \alpha$$

Symmetrisches Konfidenzintervall

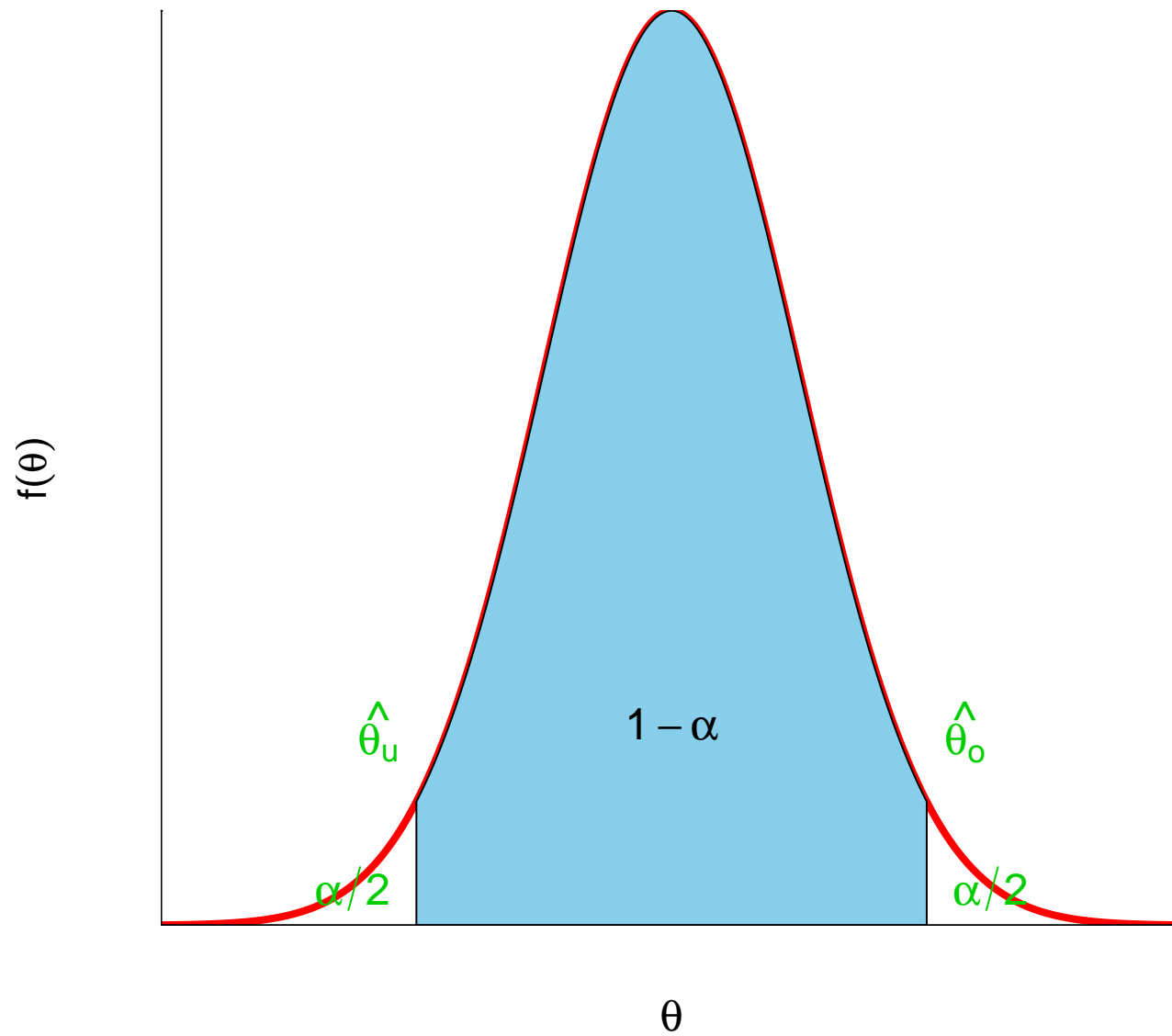
$$P\left(\theta \in [\hat{\theta}_u, \hat{\theta}_o]\right) = 1 - \alpha \quad \Leftrightarrow$$

$$P\left(\hat{\theta}_u \leq \theta \leq \hat{\theta}_o\right) = 1 - \alpha \quad \Leftarrow$$

$$P\left(\theta \leq \hat{\theta}_u\right) = \frac{\alpha}{2} \quad \text{und} \quad P\left(\theta \geq \hat{\theta}_o\right) = \frac{\alpha}{2} \quad \Leftrightarrow$$

$$P\left(\theta \leq \hat{\theta}_u\right) = \frac{\alpha}{2} \quad \text{und} \quad P\left(\theta \leq \hat{\theta}_o\right) = 1 - \frac{\alpha}{2} \quad (5.1)$$

D.h. $\hat{\theta}_u$ und $\hat{\theta}_o$ sind die $\frac{\alpha}{2}$ bzw. $1 - \frac{\alpha}{2}$ Quantile der Verteilung.



$\hat{\theta}_u$ und $\hat{\theta}_o$ sind die $\frac{\alpha}{2}$ bzw. $1 - \frac{\alpha}{2}$ Quantile der Verteilung.

Asymmetrisches Konfidenzintervall

$$P\left(\theta \in [\hat{\theta}_u, \hat{\theta}_o]\right) = 1 - \alpha \quad \Leftrightarrow$$

$$P\left(\hat{\theta}_u \leq \theta \leq \hat{\theta}_o\right) = 1 - \alpha \quad \Leftrightarrow \quad \text{wenn} \quad \alpha_1 + \alpha_2 = \alpha$$

$$P\left(\theta \leq \hat{\theta}_u\right) = \alpha_1 \quad \text{und} \quad P\left(\theta \geq \hat{\theta}_o\right) = \alpha_2 \quad \Leftrightarrow$$

$$P\left(\theta \leq \hat{\theta}_u\right) = \alpha_1 \quad \text{und} \quad P\left(\theta \leq \hat{\theta}_o\right) = 1 - \alpha_2 \quad (5.2)$$

$\hat{\theta}_u$ und $\hat{\theta}_o$ sind die α_1 bzw. $1 - \alpha_2$ Quantile der Verteilung.

Definition: einseitiges Konfidenzintervall

Setzt man in (5.2) $\alpha_1 = 0$ oder $\alpha_2 = 0$ ein, liegen einseitige Konfidenzintervalle vor:

$$\alpha_1 = 0 \Rightarrow \alpha = \alpha_2 \Rightarrow P\left(\theta \leq \hat{\theta}_o\right) = 1 - \alpha$$

$$\alpha_2 = 0 \Rightarrow \alpha = \alpha_1 \Rightarrow P\left(\theta \geq \hat{\theta}_u\right) = 1 - \alpha$$

Die einseitigen Konfidenzintervalle werden für einseitige Hypothesen-Tests verwendet und später genauer diskutiert.

Konfidenzintervalle für den Parameter p

Modell: $X \sim B(n; p)$, konkrete Beobachtung (Realisierung): $X = k$

Gesucht sind \hat{p}_u und \hat{p}_o , so dass

$$P(p \leq \hat{p}_u) = \frac{\alpha}{2} \quad \text{und} \quad P(p \geq \hat{p}_o) = \frac{\alpha}{2}$$

\hat{p}_u und \hat{p}_o werden dann durch Lösen der folg. Gleichungen berechnet:

$$\sum_{i=k}^n \binom{n}{i} (p)^i (1-p)^{(n-i)} = \alpha/2 \Rightarrow \text{Lösung } p = \hat{p}_u$$

$$\sum_{i=0}^k \binom{n}{i} (p)^i (1-p)^{(n-i)} = \alpha/2 \Rightarrow \text{Lösung } p = \hat{p}_o$$

Siehe dazu die Funktionen `binom.test()` und `binom.confint()`

binom.confint()

```
library(binom)
binom.confint(x=10, n=100)
binom.confint(x=10, n=100, method="exact")
binom.confint(x=10, n=seq(10,100,5), method="exact")
binom.confint(x=seq(10,100,10), n=100, method="exact")
binom.confint(x=seq(10,100,10), n=100,
              method="exact", conf.level=0.90)
```

Statistische Tests

- 1 Binomialtest
- 2 t-Test (Mittelwertvergleich)
 - 2.1 Der Einstichproben t-Test
 - 2.2 Der Zweistichproben t-Test (für gepaarte und ungepaarte Stichproben)
- 3 Multiple Test
- 4 F-Test (Varianzvergleich)
- 5 Anpassungstests
- 6 Testen auf Unabhängigkeit
 - 6.1 χ^2 -Test auf Unabhängigkeit
 - 6.2 Korrelationstest auf Unabhängigkeit
- 7 Varianzanalyse (Mittelwertvergleich $n > 2$)
- 8 Regressionsanalyse

Einstichproben t -Test

Es gibt drei verschiedene Fälle im Bezug auf der Hypothesenstellung:

Fall	H_0	H_1
Zweiseitig	$\mu = \mu_0$	$\mu \neq \mu_0$
Einseitig	$\mu \leq \mu_0$	$\mu > \mu_0$
Einseitig	$\mu \geq \mu_0$	$\mu < \mu_0$

und zwei verschiedene Fälle hinsichtlich des Parameters σ^2 :

- ▶ σ unbekannt (üblich)
- ▶ σ bekannt

Einstichproben t -Test: Grundlegender Gedanke

Man berechnet den Mittelwert der Stichprobe \bar{x} . Die Differenz dieser Größe und μ_0 sollte unter der Nullhypothese *nahe bei Null* sein.

Um die aufgestellte Hypothese beurteilen zu können, betrachtet man die Test-Statistik T

$$T = \frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}} \text{ wenn die Standardabweichung } \sigma \text{ unbekannt ist,}$$

$$T = \frac{\bar{X} - \mu_0}{\frac{\sigma}{\sqrt{n}}} \text{ wenn die Standardabweichung } \sigma \text{ bekannt ist.}$$

a) Zweiseitiger Test mit σ^2 unbekannt

Sei x_1, x_2, \dots, x_n eine Stichprobe n unabhängiger Beobachtungen einer $N(\mu, \sigma^2)$ -verteilten ZV mit unbekanntem μ und σ . Die Null- und Alternativhypothesen lauten:

$$H_0 : \mu = \mu_0 \quad \text{vs.} \quad H_1 : \mu \neq \mu_0$$

$$\text{Schätzer für } \mu : \bar{X} = \frac{X_1 + X_2 + \dots + X_n}{n}$$

Unter H_0 ist die Test-Statistik T t_{n-1} -verteilt:

$$T = \frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}} \sim t_{n-1}$$

Das zweiseitige $(1 - \alpha)$ -Konfidenzintervall der t_{n-1} -Verteilung berechnet sich aus:

$$\left[t_{(n-1), \frac{\alpha}{2}}, t_{(n-1), (1 - \frac{\alpha}{2})} \right] = \left[-t_{(n-1), (1 - \frac{\alpha}{2})}, t_{(n-1), (1 - \frac{\alpha}{2})} \right] : B_{H_0}$$

a) Zweiseitiger Test mit σ^2 unbekannt

Berechne $t = \frac{\bar{x} - \mu_0}{s} \sqrt{n}$ wobei $s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$ (5.3)

- Falls t in $\left[-t_{(n-1), (1-\frac{\alpha}{2})}, t_{(n-1), (1-\frac{\alpha}{2})} \right]$ liegt: H_0 beibehalten.
- sonst: H_0 zugunsten von H_1 verwerfen.

Anders formuliert:

- Falls $|t| \leq t_{(n-1), (1-\frac{\alpha}{2})}$: H_0 beibehalten.
- Falls $|t| > t_{(n-1), (1-\frac{\alpha}{2})}$: H_0 zugunsten von H_1 verwerfen.

Beispiel: Einstichproben t -Test

Man nehme an die Höhe von Roggenpflanzen sei normalverteilt. Mittels einer Studie soll die Hypothese getestet werden, dass der Mittelwert in der Grundgesamtheit $\mu_0 = 1\text{m}$ betrage. Es wurde folgende Stichprobe vom Umfang $n = 10$ entnommen.

1.02, 1.03, 1.10, 1.15, 0.99, 1.04, 1.06, 0.98, 0.97, 1.05

Wie kann man die aufgestellte Hypothese beurteilen?

Wir setzen das Signifikanzniveau $\alpha = 0.05$ fest und berechnen den Mittelwert und die Standardabweichung.

$$\bar{x} = 1.039, \quad s = 0.05547$$

$$t = \frac{\bar{x} - \mu_0}{s} \sqrt{n} = \frac{1.039 - 1}{0.05547} \sqrt{10} = 2.223343$$

$$t_{(n-1), (1-\frac{\alpha}{2})} = t_{9, 0.975} = 2.262157$$

$$|t| = 2.22 < 2.26 = t_{9, 0.975} \Rightarrow H_0 \text{ beibehalten}$$

Beispiel: Einstichproben t -Test

Berechnung in :

```
roggen <- c(1.02, 1.03, 1.10, 1.15, 0.99,  
            1.04, 1.06, 0.98, 0.97, 1.05)  
t.test(roggen, mu= 1,  
       alternative="two.sided", conf.level=0.95 )
```

One Sample t-test

data: roggen

$t = 2.2234$, $df = 9$, **p-value = 0.05326**

alternative hypothesis: true mean is not equal to 1

95 percent confidence interval: 0.9993208 1.0786792

sample estimates:

mean of x

1.039

p-value = 0.05326 $> \alpha = 0.05 \Rightarrow H_0$ beibehalten!

t.test()

```
t.test(x, y=NULL, alternative, mu=0, paired=F,  
conf.level=0.95,...)
```

x: (nicht-leerer) Vektor von Datenwerten vom Typ numeric

y: optionaler (nicht-leerer) Vektor von Datenwerten vom Typ numeric

alternative : Zeichenkette, spezifiziert die Alternativhypothese, entweder "two.sided" (default), "greater" oder "less". Anfangsbuchstabe reicht aus.

mu: Zahlenwert, wahrer (H_0) Erwartungswert (bzw. Differenz von Erwartungswerten beim two sample test)

(Hinweis: "greater" bedeutet $H_1 : \mu_x - \mu_y > mu$)

paired=F: logischer Wert, gibt an ob paired t-test oder nicht.

...

```
t.test(formula, data,...)
```

formula: Formel der Form lhs ~ rhs, mit lhs Variable von Datenwerten vom Typ numeric und rhs ein Faktor mit zwei Levels für die entsprechenden Gruppen.

data: optionale Matrix oder data.frame mit Variables aus formula. Defaultmäßig werden Variables aus environment(formula) verwendet.

KI für den Parameter μ (einer Normalverteilung)

```
roggen <- c(1.02, 1.03, 1.10, 1.15, 0.99, 1.04,  
            1.06, 0.98, 0.97, 1.05)  
t.test(roggen, mu= 1,  
       alternative="two.sided", conf.level=0.95)$conf.int
```

0.9993208 1.0786792

b) Zweiseitiger Test mit σ^2 **bekannt** (Gauß-Test)

- ▶ Es seien x_1, x_2, \dots, x_n eine Stichprobe von n unabhängigen Beobachtungen einer $N(\mu, \sigma^2)$ -verteilten ZV mit unbekanntem μ und bekannter σ . Die Null- und Alternativhypothesen lauten:
- ▶ a) $H_0 : \mu = \mu_0$ vs. $H_1 : \mu \neq \mu_0$
- ▶ Schätzer für μ : $\bar{X} = \frac{X_1 + X_2 + \dots + X_n}{n}$
- ▶ Unter H_0 ist T , die Test-Statistik, standardnormalverteilt:

$$T = \frac{\bar{X} - \mu_0}{\frac{\sigma}{\sqrt{n}}} \sim N(0, 1)$$

- ▶ Berechnen wir das zweiseitige $(1 - \alpha)$ -Konfidenzintervall der $N(0, 1)$ -Verteilung: (Z_α bezeichne das α -Quantil der $N(0, 1)$ -Verteilung)

$$\left[Z_{\frac{\alpha}{2}}, Z_{1-\frac{\alpha}{2}} \right] = \left[-Z_{1-\frac{\alpha}{2}}, Z_{1-\frac{\alpha}{2}} \right] : B_{H_0}$$

b) Zweiseitiger Test mit σ^2 **bekannt** (Gauß-Test)

Nicht in  implementiert, aber straightforward.

z.B.: Konfidenzintervall für \bar{X} bei gegebenem n , μ_0 und bekanntem σ :

```
alpha <- 0.05 # (zweiseitig):  
unten <- qnorm(alpha/2, mean=mu0, sd=(sigma) / (sqrt(n)))  
oben <- qnorm(1-alpha/2, mean=mu0, sd=(sigma) / (sqrt(n)))
```

$$B_{H_0} = [unten, oben]$$

Liegt $\bar{x} \in B_{H_0} : H_0$ beibehalten.

Zweistichproben t -Test für gepaarte Stichproben

Werden in einem Zweistichproben-Problem 2 Merkmale am selben Merkmalsträger erhoben oder ein Merkmal unter 2 Bedingungen erfasst, so liegen gepaarte Beobachtungen vor.

Beispiel: Untersuchung der Wirksamkeit eines fiebersenkenden Medikaments

In einer Studie mit einer zufällig-ausgewählten Patientengruppe hat man 50 Patienten, bei denen man die Körpertemperatur jeweils vor (x) und nach der Behandlung (y) misst.

x_1, x_2, \dots, x_{50} bzw. y_1, y_2, \dots, y_{50} die beobachtete Werte

Definiere $D_i := X_i - Y_i$ und $\mu := E(D_i)$. Die Null- und Alternativhypothesen lauten:

$$H_0 : \mu \leq 0 \quad \text{vs.} \quad H_1 : \mu > 0$$

H_0 : die Körpertemperatur wird nicht gesenkt!

t -Test für gepaarte Stichproben

a) $H_0 : \mu = 0$ vs. $H_1 : \mu \neq 0$

b) $H_0 : \mu \geq 0$ vs. $H_1 : \mu < 0$

c) $H_0 : \mu \leq 0$ vs. $H_1 : \mu > 0$

Berechne $t = \frac{\bar{d} - 0}{s} \sqrt{n}$ wobei $s^2 = \frac{1}{n-1} \sum_{i=1}^n (d_i - \bar{d})^2$ (5.4)

a) Falls $|t| > t_{(n-1), (1-\frac{\alpha}{2})}$: H_0 zugunsten von H_1 verwerfen.

b) Falls $t < -t_{(n-1), (1-\alpha)}$: H_0 zugunsten von H_1 verwerfen.

c) Falls $t > t_{(n-1), (1-\alpha)}$: H_0 zugunsten von H_1 verwerfen.

Beispiel: gepaarte Stichproben

Übung 5.5

Ziel: Vergleich von 2 Düngemitteln A und B. 10 Felder werden ausgesucht. In jedem Feld wird in einer Hälfte A und in der anderen Hälfte B eingesetzt. Erzielt wurden jeweils die Ernteerträge:

Feld	1	2	3	4	5	6	7	8	9	10
A	10	10.3	10.1	9.6	9.9	10.1	10.3	9.8	10	10.1
B	9.8	9.7	9.5	9.4	9.6	9.9	9.8	9.4	9.7	9.9
D	0.2	0.6	0.6	0.2	0.3	0.2	0.5	0.4	0.3	0.2

Fragestellung: gibt es einen (zu $\alpha = 0.05$) signifikanten Unterschied ?

Forts. Übung 5.5

1) Hypothesen definieren: $H_0 : \mu = 0$ vs. , $H_1 : \mu \neq 0$ (mit $\alpha = 0.05$)

2) Berechne $t = \frac{\bar{d} - 0}{s} \sqrt{10}$

$$\bar{d} = 0.35, s = 0.165$$

$$t = \frac{0.35 - 0}{0.165} \sqrt{10} = 6.71$$

3) $t_{(n-1), (1-\frac{\alpha}{2})}$ berechnen: $t_{9, 0.975} = 2.26$

4) $t = 6.71 > 2.26$: H_0 zugunsten von H_1 verwerfen.

Forts. Übung 5.5

Berechnung in  R:

```
feld1 <- c(10,10.3,10.1,9.6,9.9,10.1,10.3,9.8,10,10.1)
feld2 <- c(9.8,9.7,9.5,9.4,9.6,9.9,9.8,9.4,9.7,9.9)
duenge.p <- matrix(c(feld1,feld2),ncol=2)
t.test(duenge.p[,1], duenge.p[,2], alternative="two.sided",
       conf.level=.95, paired=TRUE )
```

Paired t-test

data: duenge.p[, 1] and duenge.p[, 2]

t = 6.7082, df = 9, p-value = 8.771e-05

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

0.2319721 0.4680279

sample estimates:

mean of the differences

0.35

p-value = $8.771e - 05 < \alpha = 0.05 \Rightarrow H_0$ zugunsten von H_1 verwerfen!

Forts. Übung 5.5

Alternative: t.test(formula,data)

```
b <- c(rep("feld1",10), rep("feld2",10))  
dungen <- data.frame(var1=c(feld1,feld2), var2=b)  
boxplot(var1~var2, data = dungen, ylab = "Einträge")  
t.test(formula=var1~var2, data=dungen, paired=TRUE )
```

Paired t-test

data: var1 by var2

t = 6.7082, df = 9, p-value = 8.771e-05

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

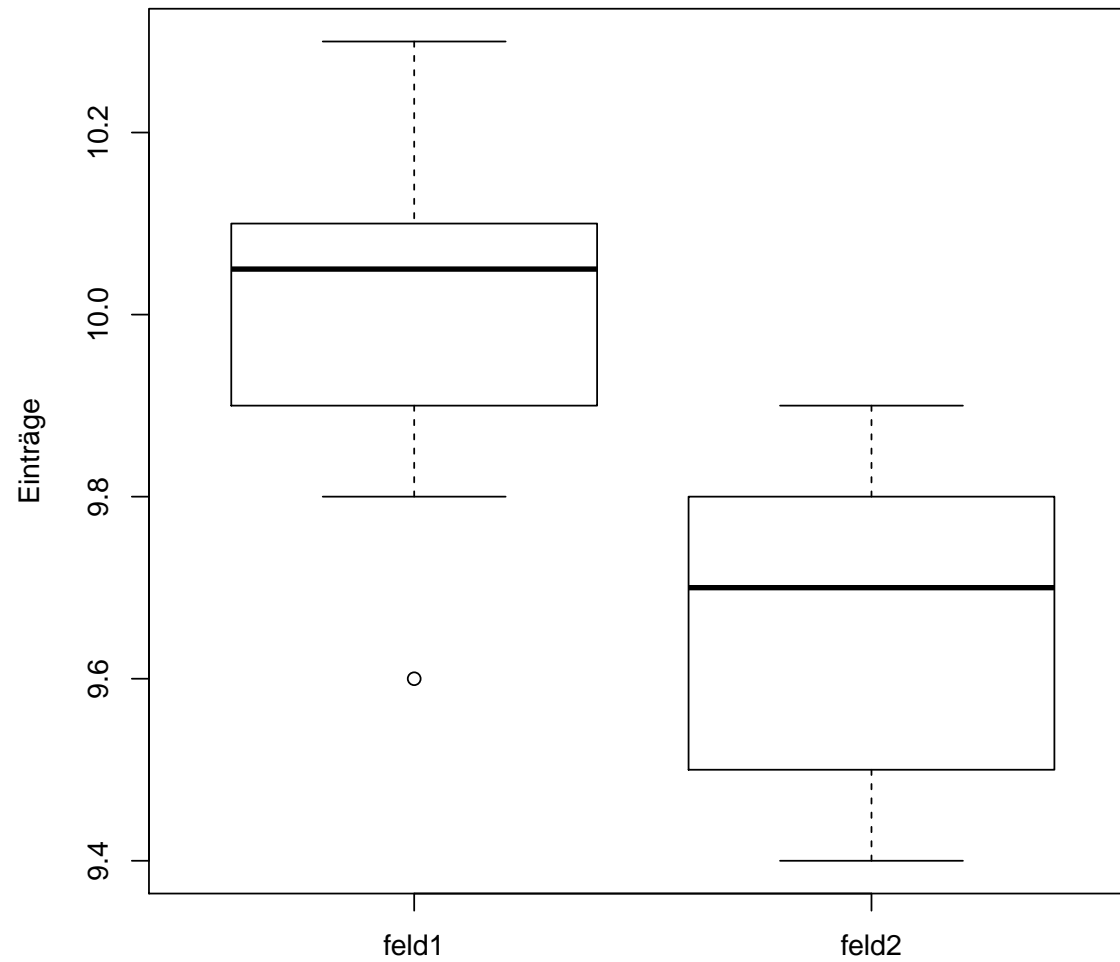
0.2319721 0.4680279

sample estimates:

mean of the differences

0.35

Forts. Übung 5.5



t -Test für ungepaarte Stichproben

Wünschenswert ist jede Versuchseinheit jeweils beiden Versuchsbedingungen zu unterwerfen.

Dies ist aber nicht immer möglich. In diesem Fall ist eine Paarbildung nicht möglich und man spricht von einer ungepaarten Stichprobe.

Ungepaarte Stichproben, gleiche Varianz

- 1) x_1, x_2, \dots, x_n bzw. y_1, y_2, \dots, y_m die beobachtete Werte als Realisierung von X_1, X_2, \dots, X_n (i.i.d.) bzw. Y_1, Y_2, \dots, Y_m (i.i.d.).
- 2) Die ZVs in beiden Gruppen sind unabhängig.
- 3) $X_i \sim N(\mu_1, \sigma_1^2)$ und $Y_i \sim N(\mu_2, \sigma_2^2)$
- 4) Die möglichen Hypothesen sind:

$$H_0 : \mu_1 = \mu_2 \quad \text{vs.} \quad H_1 : \mu_1 \neq \mu_2$$

$$H_0 : \mu_1 \geq \mu_2 \quad \text{vs.} \quad H_1 : \mu_1 < \mu_2$$

$$H_0 : \mu_1 \leq \mu_2 \quad \text{vs.} \quad H_1 : \mu_1 > \mu_2$$

Ungepaarte Stichproben, gleiche Varianz

Annahme: $\sigma_1 = \sigma_2$. Man zeigt, dass unter H_0 die folgende Testgröße T t -verteilt ist mit $n + m - 2$ Freiheitsgraden.

$$T = \frac{\bar{X} - \bar{Y}}{S_{pool} \sqrt{\frac{1}{n} + \frac{1}{m}}} = \frac{\bar{X} - \bar{Y}}{S_{pool}} \sqrt{\frac{mn}{m+n}} \sim t_{n+m-2} \text{ mit}$$
$$S_{pool} = \sqrt{\frac{(n-1)S_1^2 + (m-1)S_2^2}{n+m-2}} \quad (5.5)$$

wobei

$$S_1^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2, \quad S_2^2 = \frac{1}{m-1} \sum_{i=1}^m (Y_i - \bar{Y})^2$$

Ungepaarte Stichproben, gleiche Varianz

a) $H_0 : \mu_1 = \mu_2$ vs. $H_1 : \mu_1 \neq \mu_2$

b) $H_0 : \mu_1 \geq \mu_2$ vs. $H_1 : \mu_1 < \mu_2$

c) $H_0 : \mu_1 \leq \mu_2$ vs. $H_1 : \mu_1 > \mu_2$

Berechne $t = \frac{\bar{x} - \bar{y}}{s_{pool}} \sqrt{\frac{mn}{m+n}}$ mit $s_{pool} = \sqrt{\frac{(n-1)s_1^2 + (m-1)s_2^2}{n+m-2}}$

(5.6)

a) Falls $|t| > t_{(n+m-2), (1-\frac{\alpha}{2})}$: H_0 zugunsten von H_1 verwerfen.

b) Falls $t < -t_{(n+m-2), (1-\alpha)}$: H_0 zugunsten von H_1 verwerfen.

c) Falls $t > t_{(n+m-2), (1-\alpha)}$: H_0 zugunsten von H_1 verwerfen.

Beispiel: t -Test für ungepaarte Stichproben

Übung 5.6

Ziel: Vergleich von 2 Düngemitteln A und B. Auf 8 Feldern wird Mittel A und auf 6 Feldern Mittel B eingesetzt. Erträge:

Feld	1	2	3	4	5	6	7	8	m	sd
A	10	10.3	10.1	9.6	9.9	10.1	10.3	9.8	10	0.24
B	9.8	10.7	9.7	10.4	10.6	10.9	-	-	10.3	0.49

Gibt es einen zum Signifikanzniveau $\alpha = 0.05$ signifikanten Unterschied?

Forts. Übung 5.6

1) Hypothesen definieren: $H_0 : \mu_1 = \mu_2$ vs. $H_1 : \mu \neq \mu_2$ mit $\alpha = 0.05$

$$\begin{aligned} 2) \quad s_{pool} &= \sqrt{\frac{(n-1)s_1^2 + (m-1)s_2^2}{n+m-2}} = \\ &= \sqrt{\frac{(8-1)0.24^2 + (6-1)0.49^2}{8+6-2}} = 0.368 \end{aligned}$$

$$\begin{aligned} 3) \quad |t| &= \left| \frac{\bar{x} - \bar{y}}{s_{pool}} \sqrt{\frac{mn}{m+n}} \right| \\ &= \left| \frac{10.35 - 10.01}{0.368} \sqrt{\frac{6 \cdot 8}{8+6}} \right| = 1.685 \end{aligned}$$

4) $t_{(n+m-2), (1-\frac{\alpha}{2})}$ berechnen: $t_{12, 0.975} = 2.18$

5) $|t| = 1.685 < 2.18$: H_0 beibehalten. (mit Annahme: $\sigma_1 = \sigma_2$)

Forts. Übung 5.6

```
feld1 <- c(10,10.3,10.1,9.6,9.9,10.1,10.3,9.8)
feld2 <- c(9.8,10.7,9.7,10.4,10.6,10.9)
duenge.up <- data.frame(var1=c(feld1,feld2),
                        var2=c(rep(1,8),rep(2,6)))
t.test(duenge.up[,1] ~ duenge.up[,2],
       alternative="two.sided", conf.level=.95,
       paired=FALSE, var.equal=T)
```

Two Sample t-test

data: duenge.up[, 1] and duenge.up[, 2]

t = -1.6989, df = 12, p-value = 0.1151

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-0.77034488 0.09534488

sample estimates:

mean in group 1 mean in group 2

10.0125 10.3500

p-value = 0.1151 > $\alpha = 0.05 \Rightarrow H_0$ beibehalten!

Gepaarter und ungepaarter t -Test

Wir betrachten die folgenden Datensätze:

```
feld1 <- c(10, 10.3, 10.1, 9.6, 9.9, 10.1, 10.3, 9.8, 10, 10.1)
feld2 <- c(9.8, 9.7, 9.5, 9.4, 9.6, 9.9, 9.8, 9.4, 9.7, 9.9)
```

Wir wenden den ungepaarten und den gepaarten t -Test auf die Daten an und vergleichen die p -Werte.

```
duenge.p <- matrix(c(feld1, feld2), ncol=2)
t.test(duenge.p[,1], duenge.p[,2],
       alternative="t", paired=F)$p.value
t.test(duenge.p[,1], duenge.p[,2], , alternative="t",
       paired=T)$p.value
```

Gepaarter und ungepaarter t -Test

Vergleicht man den p -Wert aus dem t -Test für die ungepaarte Stichprobe mit dem p -Wert für die gepaarte Stichprobe

$p\text{-value} = 0.001157$: für die ungepaarte Stichprobe
 $p\text{-value} = 8.771e-05$: für die gepaarte Stichprobe,

so stellt man fest, dass die Schärfe (*Power*) des t -Tests für die gepaarte Stichprobe höher ist.

Liegen die Daten gepaart vor, soll der gepaarte t -Test verwendet werden. Der Vorteil ist, dass hierbei der Einfluss der individuellen Unterschiede eliminiert wird. Wenn die Daten wirklich paarweise vorliegen ist es auf jeden Fall vorteilhaft, diese Information zu verwenden.

Gepaarter und ungepaarter t -Test

```
data1 <- c(1, 2, 3, 4)
```

```
data2 <- c(2, 3, 4, 6)
```

```
t.test(data1, data2, paired=F, var.equal = F)$p.value
```

0.2903

```
t.test(data1, data2, paired=T)$p.value
```

0.01539

Annahme: $\sigma_1 \neq \sigma_2$. Man zeigt, dass unter H_0 die Testgröße T

$$T = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{S_1^2}{n} + \frac{S_2^2}{m}}} \sim t_g$$

approximativ t -verteilt ist mit g Freiheitsgraden, wobei

$$S_1^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2, \quad S_2^2 = \frac{1}{m-1} \sum_{i=1}^m (Y_i - \bar{Y})^2$$

und g die korrigierte Anzahl von Freiheitsgraden ist. Abrunden von g_1 ,

$$g_1 = \frac{\left(\frac{s_1^2}{n} + \frac{s_2^2}{m}\right)^2}{\frac{s_1^4}{n^2(n-1)} + \frac{s_2^4}{m^2(m-1)}} \quad (5.7)$$

auf die nächste (kleinere) ganze Zahl ergibt dann g .

Ungepaarte Stichprobe mit **ungleicher** Varianz (=Welch-test)

- a) $H_0 : \mu_1 = \mu_2$ vs. $H_1 : \mu_1 \neq \mu_2$
- b) $H_0 : \mu_1 \geq \mu_2$ vs. $H_1 : \mu_1 < \mu_2$
- c) $H_0 : \mu_1 \leq \mu_2$ vs. $H_1 : \mu_1 > \mu_2$

Berechne $t = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{s_1^2}{n} + \frac{s_2^2}{m}}}$ und (5.8)

g aus (5.7).

- a) Falls $|t| > t_{g, (1-\frac{\alpha}{2})}$: H_0 zugunsten von H_1 verwerfen.
- b) Falls $t < -t_{g, (1-\alpha)}$: H_0 zugunsten von H_1 verwerfen.
- c) Falls $t > t_{g, (1-\alpha)}$: H_0 zugunsten von H_1 verwerfen.

Feld	1	2	3	4	5	6	7	8	m	sd
A	10	10.3	10.1	9.6	9.9	10.1	10.3	9.8	10	0.24
B	9.8	10.7	9.7	10.4	10.6	10.9	-	-	10.3	0.49

Gibt es einen zum Niveau $\alpha = 0.05$ signifikanten Unterschied?

Welch-Test: $\sigma_1^2 \neq \sigma_2^2$

- 1) Hypothesen definieren: $H_0 : \mu_1 = \mu_2$ vs. $H_1 : \mu \neq \mu_2$ mit $\alpha = 0.05$
- 2) $\bar{x} = 10.35, \bar{y} = 10.01, n = 8, m = 6, s_1 = 0.24, s_2 = 0.49$
- 3) $t = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{s_1^2}{n} + \frac{s_2^2}{m}}} = \frac{10.35 - 10.01}{\sqrt{\frac{0.24^2}{8} + \frac{0.49^2}{6}}} = -1.55$
- 4) $g_1 = \frac{\left(\frac{s_1^2}{n} + \frac{s_2^2}{m}\right)^2}{\frac{s_1^4}{n^2(n-1)} + \frac{s_2^4}{m^2(m-1)}} = \frac{\left(\frac{0.24^2}{8} + \frac{0.49^2}{6}\right)^2}{\frac{0.24^4}{8^2(8-1)} + \frac{0.49^4}{6^2(6-1)}} = 6.8$
- 5) **Abrunden** von 6.8 auf die nächste (kleinere) Zahl liefert $g = 6$.
- 6) $|t| = 1.55 < t_{g, (1-\alpha/2)} = t_{6, (0.975)} = 2.45$
- 7) H_0 beibehalten. (ohne Annahme: $\sigma_1 = \sigma_2$)

Berechnung in **R**:

```
t.test(duenge.up[,1]~duenge.up[,2],  
       alternative="two.sided",  
       conf.level=.95, var.equal=F )
```

Welch Two Sample t-test

data: duenge.up[, 1] and duenge.up[, 2]

t = -1.5437, df = 6.807, **p-value = 0.1678**

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-0.8574686 0.1824686

sample estimates:

mean in group 1 mean in group 2

10.0125 10.3500

p-value = 0.1678 > $\alpha = 0.05 \Rightarrow H_0$ beibehalten!

Einseitige Alternativhypothese:

$$H_0 : \mu = \mu_0 \quad \text{vs.} \quad H_1 : \mu = \mu_1 > \mu_0$$

1. Es wird mit vorgegebenem Fehler 1. Art α und n die zu bewertende Differenz (Effekt) $\delta = \mu_1 - \mu_0$ festgelegt. Die Teststatistik T ist definiert als:

$$T = \frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}}$$

Die Power berechnet man aus den beiden Gleichungen:

$$i) \quad P(T > t_{cr} | H_0) = \alpha \text{ und}$$

$$ii) \quad P(T > t_{cr} | H_1) = 1 - \beta = \text{power}$$

Poweranalyse: Einstichproben- t -Test

$$T = \frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}} \Big| H_0 \sim t_{n-1}$$

Problem in ii):

$$T = \frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}} \Big| H_1 \not\sim t_{n-1}, \text{ sondern}$$

$$T_1 := \frac{\bar{X} - \mu_1}{\frac{S}{\sqrt{n}}} \Big| H_1 \sim t_{n-1}$$

Poweranalyse: Einstichproben- t -Test

Aus i) berechnet man t_{cr} :

$$P(T > t_{cr} | H_0) = P\left(\frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}} > t_{cr} | H_0\right) = \alpha \Rightarrow P\left(\frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}} \leq t_{cr} | H_0\right) = 1 - \alpha$$

Wir wissen $\frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}} | H_0 \sim t_{n-1} \Rightarrow$

$$t_{cr} = t_{(n-1, 1-\alpha)} \tag{5.9}$$

Poweranalyse: Einstichproben- t -Test

und mit ii) berechnen wir die Power.

1. Lösung (nicht ganz korrekt):

$$T_1 := \frac{\bar{X} - \mu_1}{\frac{S}{\sqrt{n}}} \Big| H_1 \sim t_{n-1} \text{ d.h.}$$

$$1 - \beta = P\left(\frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}} > t_{cr} \mid H_1\right) = P\left(\frac{\bar{X} - \mu_1}{\frac{S}{\sqrt{n}}} + \frac{\mu_1 - \mu_0}{\frac{S}{\sqrt{n}}} > t_{cr} \mid H_1\right) \approx$$

$$P\left(\frac{\bar{X} - \mu_1}{\frac{S}{\sqrt{n}}} > t_{cr} - \frac{\mu_1 - \mu_0}{\frac{\textcolor{red}{s}}{\sqrt{n}}} \mid H_1\right) =$$

$$1 - P\left(T_1 \leq t_{cr} - \frac{\mu_1 - \mu_0}{\frac{s}{\sqrt{n}}} \mid H_1\right) \Rightarrow$$

$$1 - \beta = 1 - P\left(T_1 \leq t'_{cr}\right) \quad \text{mit } t'_{cr} = t_{cr} - \frac{\mu_1 - \mu_0}{\frac{s}{\sqrt{n}}} \quad \text{und } T_1 \sim t_{n-1}$$

(5.10)

2. Lösung: $T = \frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}} \Big| H_1 \sim ?$

$$T = \frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}} = \frac{\frac{\bar{X} - \mu_1}{\sigma} \sqrt{n} + \frac{\mu_1 - \mu_0}{\sigma} \sqrt{n}}{\frac{S}{\sigma}}$$

Daraus folgt, dass:

$$T|H_1 \sim t_{n-1,ncp}, \text{ wobei}$$

$$ncp = \frac{\mu_1 - \mu_0}{\sigma} \sqrt{n}$$

Also:

$$1 - \beta = 1 - P\left(T \leq t_{cr}\right), \quad T \sim t_{n-1,ncp} \quad (5.11)$$

Power-Berechnung: Einstichproben- t -Test, einseitig

Gegeben sind α, n, μ_0, μ_1 und σ , mit $H_1 : \mu_1 > \mu_0$ (oder $H_1 : \mu_1 < \mu_0$).
Verwende Formeln 5.9 und 5.11:

$$t_{cr} = t_{(n-1, 1-\alpha)}$$

$$Power = 1 - P(T \leq t_{cr}), \quad T \sim t_{n-1, ncp}$$

$$\text{mit : } ncp = \frac{|\mu_1 - \mu_0|}{\sigma} \sqrt{n}$$

```
tcr <- qt(p=1-alpha, df=n-1)
ncp1 <- |mu1-mu0|*sqrt(n)/sigma
power <- 1-pt(q=tcr, df=n-1, ncp=ncp1)
```

Bemerkung: für den 2-seitigen Test setzt man in der ersten Gl. $1 - \frac{\alpha}{2}$.

Fallzahlkalkulation: Einstichproben- t -Tests, einseitig

$$H_0 : \mu = \mu_0 \quad \text{vs.} \quad H_1 : \mu = \mu_1 > \mu_0 \quad \text{oder} \quad H_1 : \mu_1 < \mu_0$$

Gegeben sind $\alpha, \beta, \delta = \mu_1 - \mu_0$ und σ . **Gesucht ist n .**

Mit den Gleichungen 5.9 und 5.10 hat man **approximativ**:

$$t_{cr} = t_{(n-1, 1-\alpha)}$$

$$t'_{cr} = t_{cr} - \frac{|\mu_1 - \mu_0|}{\frac{s}{\sqrt{n}}} = t_{(n-1, \beta)}$$

$$n \geq \left\{ \frac{\left(t_{(n-1, 1-\alpha)} + t_{(n-1, 1-\beta)} \right)}{\delta} s \right\}^2 \quad (5.12)$$

Problem: Die Zahl der Freiheitsgrade der t -Verteilung ($n-1$) ist unbekannt!

Fallzahlkalkulation: Einstichproben- t -Tests, einseitig

1. Man ersetzt die Quantile der t -Verteilung durch die entsprechenden Quantile der SN-Verteilung $z_{1-\alpha}$ und $z_{1-\beta}$ in 5.12 und berechnet n_0 :

$$n_0 \geq \left\{ \frac{(Z_{(1-\alpha)} + Z_{(1-\beta)})}{\delta} s \right\}^2$$

2. Rekursionsschritt: Man setzt n_0 auf der rechten Seite in 5.12 für die Freiheitsgrade ein und führt für n eine neue Abschätzung durch.

Bemerkung 1: weil 5.10 approximativ ist, ist die Formel 5.12 auch approximativ. Um die Fallzahl genau zu berechnen, kann man Gl. 5.9 und 5.11 rekursiv benutzen.

Bemerkung 2: für den 2-seitigen Test ersetzt man $t_{n-1,1-\alpha}$ durch $t_{n-1,1-\alpha/2}$

Effektgröße beim Einstichproben- t -Test, einseitig

Wie groß muss bei einer vorgegebenen Stichprobengröße n der tatsächliche Effekt δ sein, damit der Test eine bestimmte Power erreicht?

Die Power-Analyse kann auch verwendet werden, um die minimale Effektgröße δ in einer Studie mit gegebenen n, α, β und σ zu berechnen (approximativ).

$$\delta \geq \frac{\left(t_{(n-1, 1-\alpha)} + t_{(n-1, 1-\beta)}\right)}{\sqrt{n}} s \quad (5.13)$$

Bemerkung 1: um δ genau zu berechnen, kann man Gl. 5.11 rekursiv benutzen.

Bemerkung 2: für den 2-seitigen Test ersetzt man in 5.13 $t_{n-1, 1-\alpha}$ durch $t_{n-1, 1-\alpha/2}$.

Poweranalyse: Einstichproben- t -Tests mit *power.t.test()*

$$\text{Power} = 1 - \beta = f_1(n, \alpha, \delta, \sigma)$$

$$n = f_2(\alpha, \beta, \delta, \sigma)$$

$$\delta = f_3(n, \alpha, \beta, \sigma)$$

```
power.t.test(n = NULL, delta = NULL, sd = 1,  
             sig.level = 0.05, power = NULL,  
             type=c("two.sample", "one.sample", "paired"),  
             alternative = c("two.sided", "one.sided"))
```

Von den 5 Variablen n , δ , sd , sig.level und power sind genau vier mit konkreten Werten zu nennen und eines auf NULL zu setzen. Das auf NULL gesetzte Argument wird dann auf Basis der übrigen berechnet.

Poweranalyse: Zweistichproben- t -Tests (gepaart)

Unter Verwendung der Formeln 5.9 bis 5.13 wendet man die Poweranalyse für den Einstichproben- t -Test auf die Differenz an:

$$\delta = \delta(\text{Differenz}) = E(Y - X), \quad s = sd(\text{Differenz}) = sd(Y - X)$$

Bemerkung:

$$E(Y - X) = E(Y) - E(X), \quad \text{aber } sd(Y - X) \neq sd(Y) - sd(X)$$

Man verwendet die Funktion `power.t.test()`:

```
power.t.test(n = NULL, delta = NULL, sd = 1,
             sig.level = 0.05, power = NULL,
             type=c("two.sample", "one.sample", "paired"),
             alternative = c("two.sided", "one.sided"))
```

Poweranalyse: Zweistichproben- t -Tests (ungepaart)

Analog folgt bei einer einseitigen Fragestellung (H_1) für ungepaarte 2-Stichproben- t -Tests ($n = n_1 = n_2$ und $s_1 = s_2 = s$):

$$n \geq \frac{\left(t_{(\textcolor{red}{2n-2}, 1-\alpha)} + t_{(\textcolor{red}{2n-2}, 1-\beta)}\right)^2}{\delta^2} 2s^2 \quad (5.14)$$

$$\delta \geq \frac{\left(t_{(\textcolor{red}{2n-2}, 1-\alpha)} + t_{(\textcolor{red}{2n-2}, 1-\beta)}\right)}{\sqrt{n}} s\sqrt{2} \quad (5.15)$$

1. Für den 2-seitigen Test verwende $t_{\textcolor{red}{2n-2}, 1-\alpha/2}$.
2. Ist $n_1 \neq n_2$, dann kann 5.15 verwendet werden mit dem modifizierten n : $n = \frac{2n_1n_2}{n_1 + n_2}$
3. Ist $\sigma_1 = \sigma_2$, so kann s durch die gemeinsame Standardabweichung (*pooled sd*) s_p ersetzt werden,

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{(n_1 - 1) + (n_2 - 1)}$$

Poweranalyse: Zweistichproben- t -Tests (ungepaart)

Bemerkung:

Für die Berechnung von δ , Power und Fallzahlkalkulation mit balancierten Gruppen ($n_1 = n_2$) kann man die Funktion `power.t.test()` mit der Option `type = "two.sample"` unter Berücksichtigung der Punkte 1–3 verwenden.

Für die Fallzahlkalkulation ohne balancierten Gruppen braucht man eine andere Funktion.

Fallzahlkalkulation im t -Test mit `samplesize::n.ttest`

n.ttest

berechnet die Stichprobengröße für gepaarte und ungepaarte t -Tests. Das Test-Design kann ausgeglichen (gleiche Anzahl an Beobachtung für beide Gruppen) oder nicht ausgeglichen sein. Varianz kann über beide Gruppen homogen oder heterogen sein.

```
library(samplesize)
n.ttest(power = 0.8, alpha = 0.05, mean.diff = 0.8,
        sd1 = 0.83, sd2 = 2.65, k = 2,
        design = "unpaired", fraction = "unbalanced")
```

Statistische Tests

- 1 Binomialtest
- 2 t-Test (Mittelwertvergleich)
 - 2.1 Der Einstichproben t-Test
 - 2.2 Der Zweistichproben t-Test (für gepaarte und ungepaarte Stichproben)
- 3 Multiple Test
- 4 F-Test (Varianzvergleich)
- 5 Anpassungstests
- 6 Testen auf Unabhängigkeit
 - 6.1 χ^2 -Test auf Unabhängigkeit
 - 6.2 Korrelationstest auf Unabhängigkeit
- 7 Varianzanalyse (Mittelwertvergleich $n > 2$)
- 8 Regressionsanalyse

Multiple Tests

1. Wir generieren je zwei Datensätze $data_1$ und $data_2$ mit Umfang $n = 100$ aus $N(0, 1)$ und testen sie auf Gleichheit der Mittelwerte (mit dem Testniveau $\alpha = 0.05$):

```
set.seed(1234)
data1 <- rnorm(100)
set.seed(5678)
data2 <- rnorm(100)
t.test(data1, data2, paired=F, var.equal=T, alpha=0.05)
```

p-value = 0.922

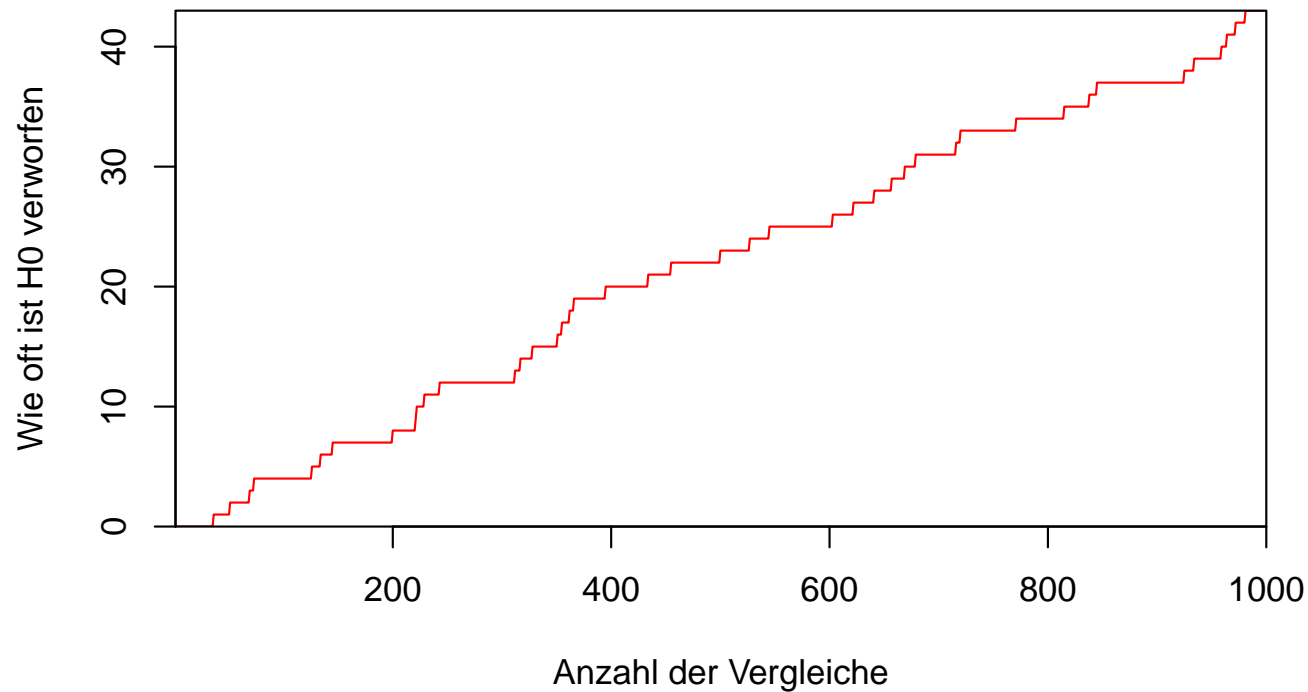
Multiple Tests

2. Jetzt simulieren wir $n_2 = 1000$ Datensätze $data_2^{(j)}$ für $j = 1, \dots, 1000$, wenden den t -Test (n_2 mal) auf $data_1$ und $data_2^{(j)}$ für $j = 1, \dots, 1000$ an (mit Testniveau $\alpha = 0.05$) und berechnen wie oft H_0 verworfen wird.

```
n1 <- 100; n2 <- 1000
data1 <- rnorm(n1)
pval <- falsch <- rep(1,n2)
for (i in 1:n2){
  data2 <- rnorm(n1)
  pval[i] <- t.test(data1, data2 ,paired=F,
                    var.equal=T, alpha=0.05)$p.value
  falsch[i] <- length(pval[pval< 0.05]) }
falsch.all <- length(pval[pval< 0.05])
falsch.all
```

43

Multiple Tests



```
plot(1:n2, falsch, typ="l", xaxs="i", yaxs="i", col=2,  
     ylab="Wie_oft_ist_H0_verworfen",  
     xlab="Anzahl_der_Vergleiche")
```


- ▶ Ein statistischer Test kann (soll) ab und zu, fälschlicherweise, zum Verwerfen der Nullhypothese führen.
- ▶ Wird mehr als eine Hypothese ($m > 1$) getestet, tritt das Problem des **multiplen Testens** auf: die W-keit, dass unter m -Tests mindestens einer falsch-signifikante Resultate liefert, ist nicht mehr gleich α .

- ▶ Ein statistischer Test kann (soll) ab und zu, fälschlicherweise, zum Verwerfen der Nullhypothese führen.
- ▶ Wird mehr als eine Hypothese ($m > 1$) getestet, tritt das Problem des **multiplen Testens** auf: die W-keit, dass unter m -Tests mindestens einer falsch-signifikante Resultate liefert, ist nicht mehr gleich α .

Multipler Test (1)

$H_{0,j}$: j-te Nullhypothese

H_0 : globale Nullhypothese = $\bigcap_{j=1}^m H_{0,j}$ vs. $H_1 = \bigcup_{j=1}^m (H_{0,j})^c$

$E_j := \{\text{j-ter Test ist signifikant}\}$

$E := \{\text{Test ist signifikant}\}$

= {mindestens einer der m Tests ist signifikant} = $\bigcup_{j=1}^m E_j$

Wird jeder Test mit dem Niveau α durchgeführt, dann hat man

$$P(E_j | H_{0,j}) = \alpha, \quad j = 1, \dots, m$$

Multipler Test (2)

gesucht ist der globale Fehler 1. Art $P(E | H_0) =: \alpha_{global}$

$$P(E_j^c | H_{0,j}) = 1 - \alpha, \quad j = 1, \dots, m$$

$$\begin{aligned} P(E | H_0) &= P\left(\bigcup_{j=1}^m E_j \mid \bigcap_{j=1}^m H_{0,j}\right) \\ &= 1 - P\left(\bigcap_{j=1}^m E_j^c \mid \bigcap_{j=1}^m H_{0,j}\right) \end{aligned}$$

und wenn die E_j -s stochastisch unabhängig sind:

$$\begin{aligned} P(E | H_0) &= 1 - \left(P(E_j^c | H_{0,j})\right)^m \\ &= 1 - (1 - \alpha)^m = \alpha_{global} \end{aligned}$$

Multipler Test (3)

- ▶ Für unsere Simulationsstudie berechnen wir den globalen α -Fehler (α_{global}) für 1000 Vergleiche:

$$\alpha_{global} = 1 - (1 - \alpha)^m = 1 - (1 - 0.05)^{1000} \approx 1$$

Die Chance einer Entscheidung für H_0 ist fast gleich Null.

- ▶ Um das Niveau α für den globalen Test zu garantieren (α_{global}), benutzt man verschiedene Methoden zur Korrektur des α -Fehlers für jeden Test, $H_{0,j}$ (α_{local}).
- ▶ Bonferroni-Korrektur: alle m Tests werden auf dem Niveau $\alpha_{local} = \alpha/m$ durchgeführt und damit hat man:

$$\alpha_{global} = P(E | H_0) = 1 - (1 - \alpha/m)^m \approx \alpha$$

Multipler Test (3)

- ▶ Für unsere Simulationsstudie berechnen wir den globalen α -Fehler (α_{global}) für 1000 Vergleiche:

$$\alpha_{global} = 1 - (1 - \alpha)^m = 1 - (1 - 0.05)^{1000} \approx 1$$

Die Chance einer Entscheidung für H_0 ist fast gleich Null.

- ▶ Um das Niveau α für den globalen Test zu garantieren (α_{global}), benutzt man verschiedene Methoden zur Korrektur des α -Fehlers für jeden Test, $H_{0,j}$ (α_{local}).
- ▶ Bonferroni-Korrektur: alle m Tests werden auf dem Niveau $\alpha_{local} = \alpha/m$ durchgeführt und damit hat man:

$$\alpha_{global} = P(E | H_0) = 1 - (1 - \alpha/m)^m \approx \alpha$$

Multipler Test (3)

- Für unsere Simulationsstudie berechnen wir den globalen α -Fehler (α_{global}) für 1000 Vergleiche:

$$\alpha_{global} = 1 - (1 - \alpha)^m = 1 - (1 - 0.05)^{1000} \approx 1$$

Die Chance einer Entscheidung für H_0 ist fast gleich Null.

- Um das Niveau α für den globalen Test zu garantieren (α_{global}), benutzt man verschiedene Methoden zur Korrektur des α -Fehlers für jeden Test, $H_{0,j}$ (α_{local}).

- **Bonferroni-Korrektur: alle m Tests werden auf dem Niveau $\alpha_{local} = \alpha/m$ durchgeführt und damit hat man:**

$$\alpha_{global} = P(E | H_0) = 1 - (1 - \alpha/m)^m \approx \alpha$$

Multipler Test (4)

- ▶ Für die Simulationsstudie gilt dann:

$$\alpha_{global} = 1 - (1 - 0.05/1000)^{1000} \approx 0.0488$$

- ▶ Damit ist garantiert, dass der α -Fehler ein bestimmtes Niveau (0.05) einhält, allerdings auf Kosten einer geringeren Power.
- ▶ *Holm*-Verfahren: der erste Test wird auf dem Niveau α/m durchgeführt; falls dieser signifikant ist, wird der zweite Test mit dem Niveau $\alpha/(m - 1)$ durchgeführt, usw.
- ▶ Die Funktion *pairwise.t.test()* ermöglicht paarweise Mittelwertvergleiche mit Korrektur für den multiplen Test.

Multipler Test (4)

- ▶ Für die Simulationsstudie gilt dann:

$$\alpha_{global} = 1 - (1 - 0.05/1000)^{1000} \approx 0.0488$$

- ▶ Damit ist garantiert, dass der α -Fehler ein bestimmtes Niveau (0.05) einhält, allerdings auf Kosten einer geringeren Power.
- ▶ *Holm*-Verfahren: der erste Test wird auf dem Niveau α/m durchgeführt; falls dieser signifikant ist, wird der zweite Test mit dem Niveau $\alpha/(m - 1)$ durchgeführt, usw.
- ▶ Die Funktion *pairwise.t.test()* ermöglicht paarweise Mittelwertvergleiche mit Korrektur für den multiplen Test.

Multipler Test (4)

- ▶ Für die Simulationsstudie gilt dann:

$$\alpha_{global} = 1 - (1 - 0.05/1000)^{1000} \approx 0.0488$$

- ▶ Damit ist garantiert, dass der α -Fehler ein bestimmtes Niveau (0.05) einhält, allerdings auf Kosten einer geringeren Power.
- ▶ *Holm-Verfahren*: der erste Test wird auf dem Niveau α/m durchgeführt; falls dieser signifikant ist, wird der zweite Test mit dem Niveau $\alpha/(m - 1)$ durchgeführt, usw.
- ▶ Die Funktion *pairwise.t.test()* ermöglicht paarweise Mittelwertvergleiche mit Korrektur für den multiplen Test.

Multipler Test (4)

- ▶ Für die Simulationsstudie gilt dann:

$$\alpha_{global} = 1 - (1 - 0.05/1000)^{1000} \approx 0.0488$$

- ▶ Damit ist garantiert, dass der α -Fehler ein bestimmtes Niveau (0.05) einhält, allerdings auf Kosten einer geringeren Power.
- ▶ *Holm*-Verfahren: der erste Test wird auf dem Niveau α/m durchgeführt; falls dieser signifikant ist, wird der zweite Test mit dem Niveau $\alpha/(m - 1)$ durchgeführt, usw.
- ▶ Die Funktion *pairwise.t.test()* ermöglicht paarweise Mittelwertvergleiche mit Korrektur für den multiplen Test.

Multipler Test: Holm-Prozedur (5)

- ▶ Führe alle Einzeltests durch, ermittle p -Werte und sortiere sie vom Kleinsten zum Größten:

$$p_1 \leq p_2 \leq \dots \leq p_m$$

- ▶ Berechne lokale α -Niveaus:

$$\alpha_1 = \frac{\alpha}{m}, \alpha_2 = \frac{\alpha}{m-1}, \dots, \alpha_m = \frac{\alpha}{1}$$

- ▶ Vergleiche die p -Werte mit den berechneten sortierten lokalen α -Niveaus (beginnend mit α_1), bis der p -Wert größer als entpr. lokale α -Niveaus ist.
- ▶ alle Nullhypothese, deren p -Wert kleiner als ihre lokalen α -Niveau waren, werden zurückgewiesen. Alle folgenden H_0 werden beibehalten.

Multipler Test: Holm-Prozedur (5)

- ▶ Führe alle Einzeltests durch, ermittle p -Werte und sortiere sie vom Kleinsten zum Größten:

$$p_1 \leq p_2 \leq \dots \leq p_m$$

- ▶ Berechne lokale α -Niveaus:

$$\alpha_1 = \frac{\alpha}{m}, \alpha_2 = \frac{\alpha}{m-1}, \dots, \alpha_m = \frac{\alpha}{1}$$

- ▶ Vergleiche die p -Werte mit den berechneten sortierten lokalen α -Niveaus (beginnend mit α_1), bis der p -Wert größer als entpr. lokale α -Niveaus ist.
- ▶ alle Nullhypothese, deren p -Wert kleiner als ihre lokalen α -Niveau waren, werden zurückgewiesen. Alle folgenden H_0 werden beibehalten.

Multipler Test: Holm-Prozedur (5)

- ▶ Führe alle Einzeltests durch, ermittle p -Werte und sortiere sie vom Kleinsten zum Größten:

$$p_1 \leq p_2 \leq \dots \leq p_m$$

- ▶ Berechne lokale α -Niveaus:

$$\alpha_1 = \frac{\alpha}{m}, \alpha_2 = \frac{\alpha}{m-1}, \dots, \alpha_m = \frac{\alpha}{1}$$

- ▶ Vergleiche die p -Werte mit den berechneten sortierten lokalen α -Niveaus (beginnend mit α_1), bis der p -Wert größer als entpr. lokale α -Niveaus ist.
- ▶ alle Nullhypothese, deren p -Wert kleiner als ihre lokalen α -Niveau waren, werden zurückgewiesen. Alle folgenden H_0 werden beibehalten.

Multipler Test: Holm-Prozedur (5)

- ▶ Führe alle Einzeltests durch, ermittle p -Werte und sortiere sie vom Kleinsten zum Größten:

$$p_1 \leq p_2 \leq \dots \leq p_m$$

- ▶ Berechne lokale α -Niveaus:

$$\alpha_1 = \frac{\alpha}{m}, \alpha_2 = \frac{\alpha}{m-1}, \dots, \alpha_m = \frac{\alpha}{1}$$

- ▶ Vergleiche die p -Werte mit den berechneten sortierten lokalen α -Niveaus (beginnend mit α_1), bis der p -Wert größer als entspr. lokale α -Niveaus ist.
- ▶ alle Nullhypothese, deren p -Wert kleiner als ihre lokalen α -Niveau waren, werden zurückgewiesen. Alle folgenden H_0 werden beibehalten.

Multipler Test: Holm-Prozedur (6)

Angenommen $\exists m_0$ s.d.

$$p_1 \leq p_2 \leq \dots \leq p_{m_0} \leq p_{m_0+1} \leq \dots \leq p_m \text{ mit}$$

$$p_1 < \frac{\alpha}{m}, p_2 < \frac{\alpha}{m-1}, \dots, p_{m_0} < \frac{\alpha}{m-m_0+1} \text{ aber}$$

$$p_{m_0+1} \geq \frac{\alpha}{m-m_0}$$

Dann werden alle entsprechenden Nullhypothesen $H_{0,1}, H_{0,2}, \dots, H_{0,m_0}$ verworfen, aber $H_{0,m_0+1}, \dots, H_{0,m}$ beibehalten.

Multiple-Test: Holm-Prozedur (7)

Anmerkung

$$p_1 < \frac{\alpha}{m} \Leftrightarrow p_1 \times m < \alpha$$

$$p_2 < \frac{\alpha}{m-1} \Leftrightarrow p_2 \times (m-1) < \alpha$$

...

$$p_{m_0} < \frac{\alpha}{m-m_0+1} \Leftrightarrow p_{m_0} \times (m-m_0+1) < \alpha$$

$$p_{m_0+1} \geq \frac{\alpha}{m-m_0} \Leftrightarrow p_{m_0+1} \times (m-m_0) \geq \alpha$$

$$p_1^{adj} := \min(p_1 \times m, 1)$$

$$p_j^{adj} := \min(p_j \times (m-j+1), 1)$$

pairwise.t.test()

Beschreibung

Erzeugt paarweise Vergleiche zwischen den Gruppen/Levels der Beobachtungen – mit Korrektur für multiples Testen.

Usage

*pairwise.t.test(x, g, p.adjust.method = p.adjust.methods,
pool.sd = TRUE, paired = FALSE, alternative = ...)*

Argumente

x Vektor der Daten (Response)

g Vektor, der die Daten gruppiert.

p.adjust.method Legt Methode zum Anpassen der *p*-Werte fest: "holm", "bonferroni", ...

pool.sd logischer Wert, gibt an ob gepoolte Standardabweichung benutzt wird (für ungepaarten *t*-Test)

paired logischer Wert, gibt an ob gepaarte *t*-Tests gewünscht sind.

Paarweiser t -Test

```
feld1 <- c(10,10.3,10.1,9.6,9.9,10.1,10.3,9.8)
feld2 <- c(9.8,10.7,9.7,10.4,10.6,10.9,11,9.8)
feld3 <- c(9.0,10.9,9.9,10.1,10.4,10.1,10.2,9.4)
duenge.up <- data.frame(var1=c(feld1,feld2,feld3),
                        var2=rep(1:3, each = 8))
pairwise.t.test(duenge.up[,1], duenge.up[,2],
                alternative="t",p.adj="bonferroni",
                pool.sd =F)
```

Pairwise comparisons using t tests with pooled SD

data: duenge.up[, 1] and duenge.up[, 2]

	1	2
2	0.36	-
3	1.00	0.64

P value adjustment method: holm

```
t.test(feld1, feld2)$p.value
t.test(feld1, feld3)$p.value
t.test(feld2, feld3)$p.value
```

Statistische Tests

- 1 Binomialtest
- 2 t-Test (Mittelwertvergleich)
 - 2.1 Der Einstichproben t-Test
 - 2.2 Der Zweistichproben t-Test (für gepaarte und ungepaarte Stichproben)
- 3 Multiple Test
- 4 F-Test (Varianzvergleich)
- 5 Anpassungstests
- 6 Testen auf Unabhängigkeit
 - 6.1 χ^2 -Test auf Unabhängigkeit
 - 6.2 Korrelationstest auf Unabhängigkeit
- 7 Varianzanalyse (Mittelwertvergleich $n > 2$)
- 8 Regressionsanalyse

Varianzvergleich: Einstichprobentest

Sei $X_i \sim N(\mu, \sigma^2)$ (i.i.d.) mit **unbekannten** μ, σ

$H_0: \sigma = \sigma_0$. Als Schätzer für die Varianz verwenden wir

$$S^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2 \Rightarrow \frac{n-1}{\sigma^2} S^2 \sim \chi_{n-1}^2$$

Falls μ bekannt ist, verwendet man den Schätzer \tilde{S}^2

$$\tilde{S}^2 = \frac{1}{n} \sum_{i=1}^n (X_i - \mu)^2 \Rightarrow \frac{n}{\sigma^2} \tilde{S}^2 \sim \chi_n^2$$

`library(TeachingDemos)`

`sigma.test(x, sigma = 1, alternative, conf.level = 0.95)`

Varianzvergleich: 2-Stichprobentest (F -Test)

Modell:

$$X_i \sim N(\mu_1, \sigma_1^2), \quad 1 \leq i \leq n \text{ (i.i.d.)}$$

$$Y_i \sim N(\mu_2, \sigma_2^2), \quad 1 \leq i \leq m \text{ (i.i.d.)}$$

- a) $H_0 : \sigma_1 = \sigma_2$ vs. $H_1 : \sigma_1 \neq \sigma_2$
- b) $H_0 : \sigma_1 \geq \sigma_2$ vs. $H_1 : \sigma_1 < \sigma_2$
- c) $H_0 : \sigma_1 \leq \sigma_2$ vs. $H_1 : \sigma_1 > \sigma_2$

Test-Statistik $T = \frac{\frac{S_x^2}{n-1}}{\frac{S_y^2}{m-1}},$ verglichen mit : $F_{(n-1),(m-1)}$ (5.16)

- a) Falls $F_{(n-1),(m-1);\frac{\alpha}{2}} \leq T \leq F_{(n-1),(m-1);1-\frac{\alpha}{2}} : H_0$ beibehalten.
- b) Falls $T \geq F_{(n-1),(m-1);\alpha} : H_0$ beibehalten.
- c) Falls $T \leq F_{(n-1),(m-1);1-\alpha} : H_0$ beibehalten.

var.test()

Führt einen F-Test aus, um Varianzen zweier normalverteilter Stichproben zu vergleichen.

```
var.test(x,y,ratio = 1,alternative,conf.level=0.95,...)
```

```
var.test(formula, data, ...)
```

x,y: Numerischer Vektor von Datenwerten, oder Objekte vom Typ *fitted linear model* (aus der Klasse "*lm*").

ratio: Verhältnis der Varianzen von *x* und *y* unter der Nullhypothese

formula: Formel der Form *lhs~rhs*, wobei *lhs* eine Variable ist, die die (numerischen) Datenwerte enthält und *rhs* ein Faktor mit zwei Levels für die zugehörige Gruppeneinteilung.

data : optionale Matrix oder Data frame ...

var.test()

```
set.seed(123)
x <- rnorm(100, m=10, sd=5)
y <- rnorm(80, m=12, sd=4)
boxplot(x, y)
var.test(x, y, alternative="g")
```

$$H_0 : \sigma_1 \leq \sigma_2 \quad vs. \quad H_1 : \sigma_1 > \sigma_2$$

Übung 5.7

Testen Sie die Erträge der 2 Feldern A und B auf die Varianzheterogenität ($H_1 : \sigma_1 \neq \sigma_2$) mit $\alpha = 0.1$.

Lösung

```
feld1 <- c(10,10.3,10.1,9.6,9.9,10.1,10.3,9.8)
feld2 <- c(9.8,10.7,9.7,10.4,10.6,10.9)
duenge.up <- data.frame(var1=c(feld1,feld2),
                        var2=c(rep(1,8),rep(2,6)))
var.test(duenge.up[,1] ~ duenge.up[,2],
         alternative="two.sided", conf.level=.90 )
```

F.test to compare two variances

data: duenge.up[, 1] by duenge.up[, 2]

F = 0.2403, num df = 7, denom df= 5 p-value = 0.08969

alternative hypothesis: true ratio of variances is not equal to 1

90 percent confidence interval:

0.04928346 0.95435631

sample estimates:

ratio of variances

0.2402998

p-value = 0.08969 < $\alpha = 0.1 \Rightarrow H_0$ verwerfen!

Lösung... etwas eleganter

Funktion *stack()* aus dem Package *utils* benutzen:

```
feld1 <- c(10,10.3,10.1,9.6,9.9,10.1,10.3,9.8)
feld2 <- c(9.8,10.7,9.7,10.4,10.6,10.9)
library(utils)
duenge.up <- stack(list("1"=feld1,"2"=feld2))
var.test(duenge.up$values ~ duenge.up$ind,
         alternative="two.sided", conf.level=.90 )
```

F.test to compare two variances

data: duenge.up[, 1] by duenge.up[, 2]

F = 0.2403, num df = 7, denom df= 5 p-value = 0.08969

alternative hypothesis: true ratio of variances is not equal to 1

90 percent confidence interval:

0.04928346 0.95435631

sample estimates:

ratio of variances

0.2402998

p-value = 0.08969 < $\alpha = 0.1 \Rightarrow H_0$ verwerfen!

Varianzhomogenitäts-Test

- ▶ Der Test auf Varianzhomogenität benötigt man bei vielen Test-Verfahren als Vortest, z.B. für Mittelwertvergleiche (*t*-Test) oder Varianzanalyse (ANOVA).
- ▶ *var.test()* vergleicht die Quotienten zweier Varianzen beruhend auf dem *F*-test.
- ▶ Für den Vergleich von mehr als zwei Varianzen kann entweder der parametrische *Bartlett-Test* (*bartlett.test()*) oder der nichtparametrische *Fligner-Test* (*fligner.test()*) angewandt werden.

Varianzhomogenitäts-Test

- ▶ Der Test auf Varianzhomogenität benötigt man bei vielen Test-Verfahren als Vortest, z.B. für Mittelwertvergleiche (*t*-Test) oder Varianzanalyse (ANOVA).
- ▶ *var.test()* vergleicht die Quotienten zweier Varianzen beruhend auf dem *F*-test.
- ▶ Für den Vergleich von mehr als zwei Varianzen kann entweder der parametrische *Bartlett-Test* (*bartlett.test()*) oder der nichtparametrische *Fligner-Test* (*fligner.test()*) angewandt werden.

Varianzhomogenitäts-Test

- ▶ Der Test auf Varianzhomogenität benötigt man bei vielen Test-Verfahren als Vortest, z.B. für Mittelwertvergleiche (*t*-Test) oder Varianzanalyse (ANOVA).
- ▶ *var.test()* vergleicht die Quotienten zweier Varianzen beruhend auf dem *F*-test.
- ▶ Für den Vergleich von mehr als zwei Varianzen kann entweder der parametrische *Bartlett-Test* (*bartlett.test()*) oder der nichtparametrische *Fligner-Test* (*fligner.test()*) angewandt werden.

5. Statistische Tests

- 1 Binomialtest
- 2 t-Test (Mittelwertvergleich)
- 3 Multiple Test
- 4 F-Test (Varianzvergleich)
- 5 Anpassungstests (*Goodness-of-fit test*)
 - a χ^2 -Anpassungstest für kategoriale Merkmale
 - b Anpassungstest für metrisch-skalierte Merkmale
 - b1 Anpassungstests auf Normalverteilung: Shapiro-Wilk-Test

Anpassungstest

Testen der Nullhypothese, ob die Stichprobe einer vorgegebenen Verteilung (z.B. Normal-, t -, Binomial,...) folgt, d.h. $H_0 : F = F_0$.

F : tatsächliche (unbekannte) Verteilung,

F_0 : vorgegebene Verteilung

Es wird getestet, ob die Abweichung der beobachteten Verteilung von der erwarteten Verteilung statistisch signifikant ist.

Voraussetzung: $X_1, X_2, \dots, X_n \stackrel{i.i.d.}{\sim} F$, F unbekannt

$$H_0 : F = F_0, \quad H_1 : F \neq F_0$$

χ^2 -Anpassungstest für kategoriale Merkmale: Beispiel

Wir würfeln 120-mal und finden die folgenden Häufigkeiten...

1	2	3	4	5	6
18	16	21	23	17	25

welche aus einer unbekannten Verteilung stammen, d.h.

$$X, X_1, X_2, \dots, X_{100} \stackrel{i.i.d.}{\sim} F$$

Sind dies Beobachtungen von fairen Würfelwürfen?

Genau dann wenn $F = F_0$ mit

$$F : P(X = j) = p_j, \quad \forall j \in \{1, 2, \dots, k\} \quad \text{hier } k = 6$$

$$F_0 : P(X = j) = p_{j,0} = 1/6, \quad \forall j \in \{1, 2, \dots, k\}$$

Testhypothesen:

$$H_0 : p_j = p_{j,0} = 1/6, \quad \forall j \in \{1, 2, \dots, k\}$$

$$H_1 : \exists j \in \{1, 2, \dots, 6\}, \quad p_j \neq p_{j,0} = 1/6$$

χ^2 -Anpassungstest zum Niveau α

Schätze p_j durch relative Häufigkeiten

$$\hat{p}_j = \frac{h_j}{n}, \quad h_j = \text{abs. Häufigkeiten}$$

und *vergleiche* die \hat{p}_j und $p_{j,0}$ mittels geeignetem Abstandsmaß:

$$X^2 = n \sum_{j=1}^k \frac{(\hat{p}_j - p_{j,0})^2}{p_{j,0}} = \sum_{j=1}^k \frac{(h_j - np_{j,0})^2}{np_{j,0}}$$

Die Berechnung der exakten Verteilung $F(x) = P(X^2 \leq x)$ ist mühsam, allerdings gilt für die asymptotische Verteilung:

$$X^2 \stackrel{asym}{\sim} \chi_{k-1}^2 \quad (\text{unter } H_0)$$

Approximation anwendbar, wenn $np_{j,0} \geq 1, \forall j \in \{1, 2, \dots, k\}$ und $np_{j,0} \geq 5$ für mindestens 80% der Klassen.

Ablehnungsbereich: $X^2 > \chi_{k-1, 1-\alpha}^2$; (wenn X^2 zu groß ist)

χ^2 -Anpassungstest zum Niveau α

Wir testen die Nullhypothese, dass der Würfel fair ist, d.h.
 $p_j = 1/6$ mit $\alpha = 0.05$:

Ereignis	1	2	3	4	5	6
h_j	18	16	21	23	17	25
$np_{j,0}$	20	20	20	20	20	20
$\frac{(h_j - np_{j,0})^2}{np_{j,0}}$	4/20	16/20	1/20	9/20	9/20	25/20

$$X^2 = \frac{4}{20} + \frac{16}{20} + \frac{1}{20} + \frac{9}{20} + \frac{9}{20} + \frac{25}{20} = 64/20 = 3.2$$

Kritischer Wert:

$$\chi^2_{6-1, 1-0.05} = \chi^2_{5, 0.95} = 11.0705$$

$$X^2 = 3.2 \leq \chi^2_{5, 0.95} = 11.0705$$

$\Rightarrow H_0$ wird nicht verworfen.

χ^2 -Anpassungstest zum Niveau α mit

```
x <- c(18 , 16 , 21 , 23 , 17 , 25)
chisq.test(x, p=rep(1/6, 6))
```

X-squared = 3.2, df = 5, p-value = 0.6692

Berechnung des p-Wertes auf Basis einer Monte Carlo Simulation mit B Wiederholungen (*replicates*):

```
chisq.test(x, p=rep(1/6, 6),
           simulate.p.value = TRUE,
           B = 20000)
```

X-squared = 3.2, df = NA, p-value = 0.6794

?chisq.test

Beschreibung

chisq.test führt chi-Quadrat Kontingenztafel- und Goodness-of-fit Tests aus.

Anwendung

```
chisq.test(x, y = NULL, correct = TRUE,  
p = rep(1/length(x), length(x)), rescale.p = FALSE,  
simulate.p.value = FALSE, B = 2000)
```

Argumente

x numerischer Vektor oder Matrix. x und y können auch beide Faktoren sein.

y numerischer Vektor; wird ignoriert, falls x eine Matrix ist.

p Vektor von Wahrscheinlichkeiten mit gleicher Länge wie x.

simulate.p.value logischer Wert, gibt an, ob p-Werte mittels Monte Carlo Simulation berechnet werden sollen.

B Integer, Anzahl der Wiederholungen im Fall der p-Wert-Bestimmung via Monte Carlo.

Übung 5.8

Der Datensatz *survey* aus dem Paket *MASS* enthält die Antworten von 237 Studierenden auf versch. Fragen. Unter anderen wurde das Rauchverhalten (Smoke) abgefragt. Das ordinal-skalierte Merkmal *Smoke* hat die Ausprägungen: *Heavy, Regul, Occas, Never*. Testen Sie die folgenden Nullhypothese:

$$c(P_{Heavy}, P_{Regul}, P_{Occas}, P_{Never}) = c(0.05, 0.10, 0.10, 0.75)$$

zum Niveau $\alpha = 0.05$.

Übung 5.8

Der Datensatz *survey* aus dem Paket *MASS* enthält die Antworten von 237 Studierenden auf versch. Fragen. Unter anderen wurde das Rauchverhalten (Smoke) abgefragt. Das ordinal-skalierte Merkmal *Smoke* hat die Ausprägungen: *Heavy, Regul, Occas, Never*. Testen Sie die folgenden Nullhypothese:

$$c(P_{Heavy}, P_{Regul}, P_{Occas}, P_{Never}) = c(0.05, 0.10, 0.10, 0.75)$$

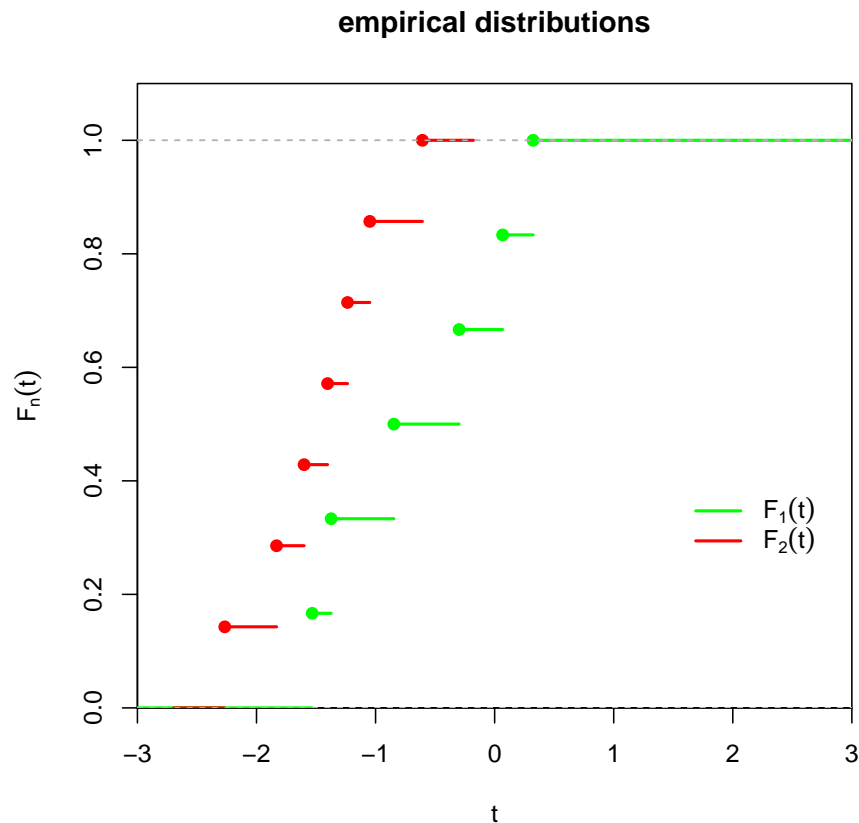
zum Niveau $\alpha = 0.05$.

```
data(survey, package="MASS")
agg <- table(survey$Smoke)
head(agg)
chisq.test(x=as.vector(agg), p=c(.05, .75, .1, .1))
```

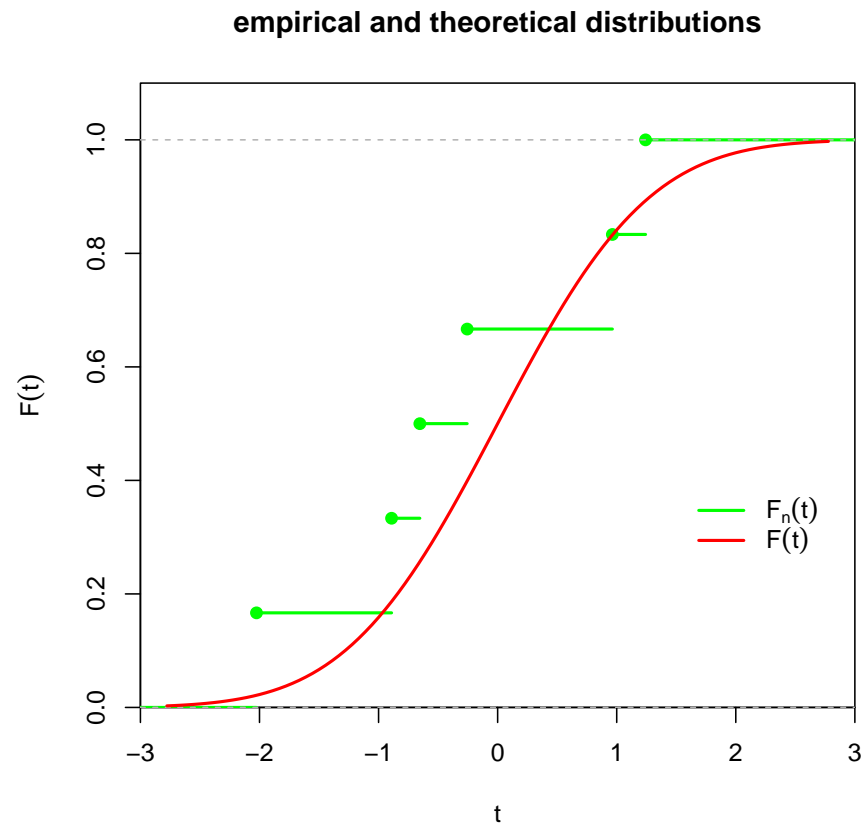
Anpassungstest für stetige Merkmale

EDF-tests (empirical distribution function tests)
basieren auf dem Vergleich von Verteilungsfunktionen.

Vergleich zweier empirischer Verteilungsfunktionen:



Vergleich einer empirischen und einer konkreten theoretischen Verteilungsfunktion:



Einseitige Hypothesen

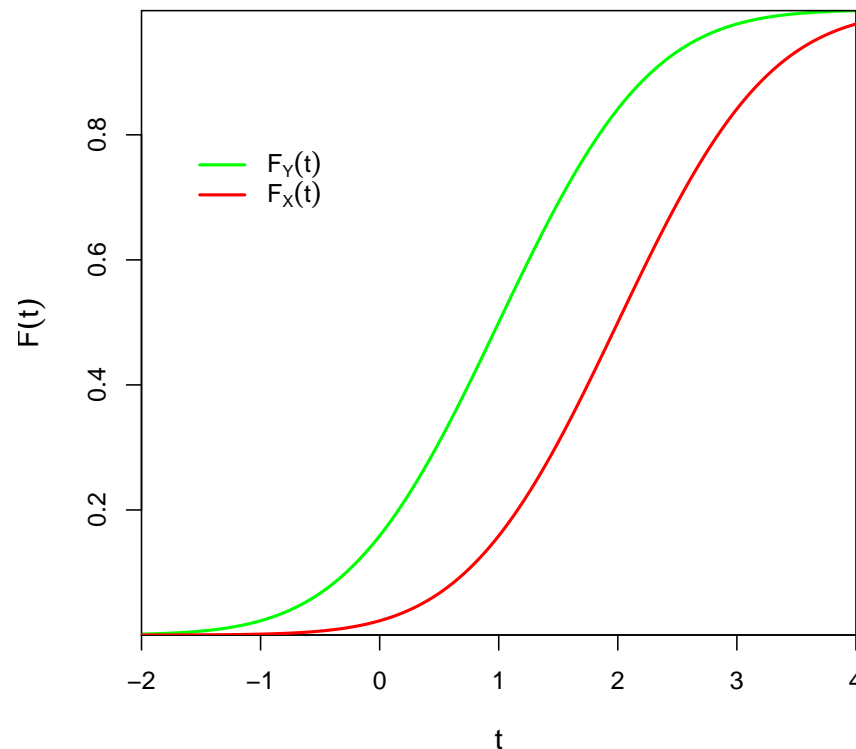
$$\mathbf{X} \stackrel{P}{\geq} \mathbf{Y}$$

$$\mathbf{F_X} \leq \mathbf{F_Y}$$

Die ZV X heisst stochastisch grösser als Y

$$P(Y > t) = 1 - F_Y(t)$$

$$P(X > t) = 1 - F_X(t)$$



$$P(X > t) \geq P(Y > t)$$

$$\Leftrightarrow F_X(t) \leq F_Y(t)$$

für alle t .

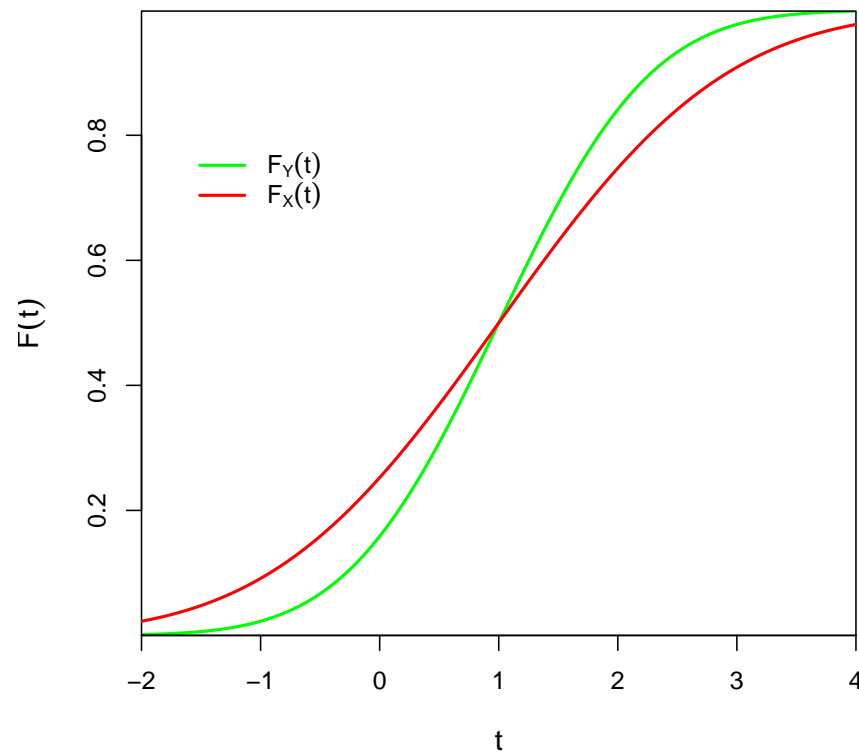
Beispiel:

$$Y \sim N(0, 1)$$

$$X \sim N(1, 1)$$

Zweiseitige Hypothesen

$$F_X \neq F_Y$$



$\exists t_0$, sodass

$$F_X(t_0) \neq F_Y(t_0)$$

hier:

$$Y \sim N(1, 1)$$

$$X \sim N(1, 1.5^2)$$

Kolmogorov-Smirnov-Test

Einstichprobenfall: die Verteilung von X , F , wird mit einer vorgegebenen Verteilung F_0 verglichen.

- | | | |
|---|-----|--------------------------------------|
| a) $H_0 : \forall t \ F(t) = F_0(t)$ | vs. | $H_1 : \exists t \ F(t) \neq F_0(t)$ |
| b) $H_0 : \forall t \ F(t) \geq F_0(t)$ | vs. | $H_1 : \exists t \ F(t) < F_0(t)$ |
| c) $H_0 : \forall t \ F(t) \leq F_0(t)$ | vs. | $H_1 : \exists t \ F(t) > F_0(t)$ |

Anstatt F wird die empirische Verteilungsfunktion F_n mit F_0 mit Hilfe der Teststatistik K_n verglichen:

- | |
|---------------------------------------|
| a) $K_n = \sup_t F_0(t) - F_n(t) $ |
| b) $K_n^+ = \sup_t (F_0(t) - F_n(t))$ |
| c) $K_n^- = \sup_t (F_n(t) - F_0(t))$ |

Ablehnungsbereich: falls K_n, K_n^+, K_n^- zu groß wird.

Kolmogorov-Smirnov-test (2)

Bemerkung 1: $F_n \xrightarrow{glm.} F$ (Fundamentalsatz der Statistik- Satz von Gliwenko-Cantelli).

Bemerkung 2: Unter H_0 gilt: die Verteilung von $K_n(K_n^+, K_n^-)$ hängt nur von n und nicht von F_0 ab (Verteilungsfreie Statistik), siehe Gibbson & Chakraborti (1992): $K_n \sim k_n$

Vergleiche den Wert der Teststatistik $K_n(K_n^+, K_n^-)$ mit der k_n -Verteilung:

a) $K_n \geq k_{n,1-\alpha} \Rightarrow H_0$ verwerfen

b) $K_n^+ \geq k_{n,1-\alpha}^+ \Rightarrow H_0$ verwerfen

c) $K_n^- \geq k_{n,1-\alpha}^- \Rightarrow H_0$ verwerfen

Bemerkung 3: $k_{n,1-\alpha}^+ = k_{n,1-\alpha}^- \approx k_{n,1-2\alpha}$ für kleine α

Bemerkung 4: $\sqrt{n}K_n \overset{asy}{\sim} k$ (k : Kolmogorov-Verteilung)

Beschreibung

Führe den ein- oder zwei-Stichproben Kolmogorov-Smirnov-Test durch.

Anwendung

ks.test(x,y,...,alternative=c("t", "l", "g"),exact = NULL)

Argumente

x ein numerischer Vektor von Datenwerten.

y entweder ein numerischer Vektor von Datenwerten, oder eine Zeichenkette die eine Verteilungsfunktion benennt oder eine Verteilungsfunktion wie z.B. pnorm. Verteilungsfunktionen müssen stetig sein.

... Parameters der durch *y* (als Zeichenkette) spezifizierten Verteilung.

exact NULL oder ein logischer Wert, der angibt ob der exakte p-Wert berechnet werden soll...

```
x <- runif(5)
ks.test(x, "punif")
ks.test(x, "pnorm")
ks.test(x, "pnorm", mean=1, sd=1)
ks.test(x, "punif", min=-1, max=1)
```

Bemerkung 5: Eine Einschränkung des *KS-Tests* besteht darin, dass die Parameter von F_0 bekannt sein sollen. Werden die Parameter aus der Stichprobe geschätzt (zusammengesetzte Nullhypothese), so erhält man die Teststatistik \hat{K}_n , deren Verteilung nicht mit der Verteilung von K_n übereinstimmt. Verwendet man für die Teststatistik \hat{K}_n die kritischen Werte von K_n , verliert man an Power (konservativer Test).

KS-Test mit Lilliefors-Korrektur

zusammengesetzte (Null-)hypothese:

$$H_0 : F \in \mathcal{F}_0 \quad \text{mit} \quad \mathcal{F}_0 = \{F_\theta | \theta \in \Theta\}$$

$$H_1 : F \notin \mathcal{F}_0$$

Lilliefors hat durch Simulationen Quantile der Verteilungen von \hat{K}_n für die Verteilungsfamilien der Normal- und Exponentialverteilung ermittelt (verteilungsgebundener Test). Der Test für die Normalverteilung kann durch die Funktion *lillie.test* (aus dem Paket *nortest*) durchgeführt werden.

```
install.packages("nortest")  
library(nortest)  
ks.test(x, "pnorm", m=mean(x), sd=sd(x))  
lillie.test(x)
```

Andere Anpassungstests

Weitere (KS-ähnliche) EDF-Tests: *Anderson-Darling*- und *Cramer-von Mises*-Tests basieren auf alternative Definitionen des Abstandsmasses zwischen F_0 und F_n .

Shapiro-Wilk-Test (verteilungsgebundener Test):

$H_0 : X$ ist normalverteilt vs. $H_1 : X$ ist nicht normalverteilt.

Bemerkung 6: In Simulationsstudien lieferte der Shapiro-Wilk-Test das schärfste Ergebnis beim Test auf Normalverteilung, gefolgt von Anderson-Darling-, Lilliefors- und Kolmogorov-Smirnov-Tests. Siehe: *Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling test, Journal of Statistical Modeling and Analytics, Vol.2, No.1, 21-33, 2011*

shapiro.test(x) (in Paketen: *stats*)

ad.test(x) , *cvm.test(x)* (in Paketen: *nortest* für Test auf Normalverteilung, *goftest* für Test auf genau spezifizierte Verteilung)

Beispiele in

```
powerfkt <- function(alpha=0.05,size=50,m=1000) {  
  library(nortest)  
  H <- matrix(logical(), ncol=5, nrow=m)  
  colnames(H)=c("KS", "AD", "CVM", "SW", "LIL")  
  for (i in 1:m){ # Testentscheidungen aufschreiben  
    set.seed(1234+i)  
    data <- runif(size,-1,1)  
    t1 <- ks.test(data,"pnorm",m=mean(data),sd=sd(data))$p.value  
    t2 <- ad.test(data)$p.value  
    t3 <- cvm.test(data)$p.value  
    t4 <- shapiro.test(data)$p.value  
    t5 <- lillie.test(data)$p.value  
    # wenn p-wert < 0.05, entscheide für alternative:  
    H[i,] <- ifelse(c(t1,t2,t3,t4,t5) < 0.05,TRUE,FALSE)  
  }  
  power <- apply(H, 2, sum)/m # Power der Tests  
  return(power)  
}  
powerfkt()
```

Übung 5.9

Von Büning und Trenkler stammt folgende Stichprobe zum Benzinverbrauch eines PKW-Typs in Litern pro 100 km, die von 10 verschiedenen Fahrzeugen des gleichen Typs bei einer Geschwindigkeit von 100 km/h ermittelt wurden:

12.4 11.8 12.9 12.6 13.0 12.5 12.0 11.5 13.2 12.8

- a) Testen Sie die Nullhypothese, dass die Daten aus einer Normalverteilung mit $\mu = E(X) = 12$ und $\sigma = 1$ stammen, mit geeigneten Tests.
- b) Testen Sie die allgemeinere Hypothese, dass die Daten aus der Familie der Normalverteilungen stammen, mit geeigneten Tests.

5. Statistische Tests

- 1 Binomialtest
- 2 t-Test (Mittelwertvergleich)
- 3 Multiple Test
- 4 F-Test (Varianzvergleich)
- 5 Anpassungstests
- 6 Testen auf Unabhängigkeit
 - 6.1 χ^2 -Test auf Unabhängigkeit
 - 6.2 Korrelationstest auf Unabhängigkeit

χ^2 -Test auf Unabhängigkeit

Der Zusammenhang von ordinal- und metrisch-skalierten Merkmalen wird durch die Korrelation und der Zusammenhang von nominal-skalierten Merkmalen durch die Kontingenz beschrieben.

S und T seien ZVen mit diskreten Werten $S \in \{s_1, \dots, s_I\}$ und $T \in \{t_1, \dots, t_J\}$. Weiter sei $N_{ij} := \#\{(S, T) = (s_i, t_j)\}$.

$\begin{matrix} \text{T} \\ \text{S} \end{matrix}$	t_1	t_2	\dots	t_J	Σ
s_1	N_{11}	N_{12}	\dots	N_{1J}	$N_{1.}$
s_2	N_{21}	N_{22}	\dots	N_{2J}	$N_{2.}$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
s_I	N_{I1}	N_{I2}	\dots	N_{IJ}	$N_{I.}$
	$N_{.1}$	$N_{.2}$	\dots	$N_{.J}$	N

Unter H_0 gilt: $P\left((S, T) = (s_i, t_j)\right) = P(S = s_i) \times P(T = t_j) \Rightarrow$

χ^2 -Test auf Unabhängigkeit (2)

$$\frac{N_{ij}}{N} \approx \frac{N_{i.}}{N} \times \frac{N_{.j}}{N} \quad \text{unter } H_0 \Rightarrow$$

$$N_{ij} \approx \frac{N_{i.} N_{.j}}{N} \quad \text{unter } H_0$$

Definiere: $\tilde{N}_{ij} := \frac{N_{i.} N_{.j}}{N}$

$$\chi^2 = \sum_{i=1}^I \sum_{j=1}^J \frac{(N_{ij} - \tilde{N}_{ij})^2}{\tilde{N}_{ij}}$$

- Unter H_0 gilt: χ^2 -Statistik $\overset{asy}{\sim} \chi^2_{(I-1) \times (J-1)}$
- Testentscheidung: H_0 verwerfen, wenn $\chi^2 > \chi^2_{1-\alpha, (I-1) \times (J-1)}$
- Für die Approximation der χ^2 -Statistik durch $\chi^2_{(I-1) \times (J-1)}$ gelten die gleichen Regeln wie für den χ^2 -Anpassungstest.

Übung 5.10

Testen Sie die Unabhängigkeit der Merkmalen *Smoke* und *Sex* im Datensatz *survey* mit dem Niveau $\alpha = 0.05$.

Übung 5.10

Testen Sie die Unabhängigkeit der Merkmalen *Smoke* und *Sex* im Datensatz *survey* mit dem Niveau $\alpha = 0.05$.

```
library(MASS)
attach(survey)
chisq.test(Smoke, Sex)
chisq.test(Smoke, Sex, simulate.p.value=T)
detach(survey)
```

X-squared = 3.55, df = 3, p-value = 0.3139

Übung 5.11

Laden Sie den Datensatz *heartatk.txt* aus *StudIP*. Testen Sie die Abhängigkeit der Merkmalen SEX und DIED mit einem Signifikanzniveau von 5%.

fisher.test()

Der exakte Fisher-Test (auf Unabhängigkeit) wird Kontingenztafeln mit geringem Stichprobenumfang verwendet.

Die folgende Tabelle zeigt die Heilungsraten in einer Studie bei Standardtherapie und einer neuen Therapie:

Therapie	Geheilt	
	ja	nein
Standard	4	16
Neu	10	8

Testen Sie die Alternativhypothese, dass die neue Therapie besser als die Standardtherapie ist (mit $\alpha = 0.05$).

```
x <- matrix(c(4, 16, 10, 8), ncol=2, byrow=T)
chisq.test(x)
fisher.test(x)
```

Statistische Tests

- 1 Binomialtest
- 2 t-Test (Mittelwertvergleich)
- 3 Multiple Test
- 4 F-Test (Varianzvergleich)
- 5 Anpassungstests
- 6.1 χ^2 -Test auf Unabhängigkeit
- 6.2 **Korrelationstest auf Unabhängigkeit**
 - Der Zusammenhang von ordinal- oder metrisch-skalierten Merkmalen wird durch die Korrelationskoeffizienten definiert.
 - 6.2.1 Korrelationstest bei 2-dimensionalen Normalverteilung
 - 6.2.2 Korrelationstest bei stetigen und ordinal-skalierten Merkmalen

Pearson-Korrelationstest

Voraussetzung:

$$(X_i, Y_i) \stackrel{i.i.d.}{\sim} \mathcal{N}\left(\mu, \Sigma\right)$$

Sei ρ_p der theoretische und $\hat{\rho}_p$ der empirische pearson-Korrelationskoeffizient von (X_i, Y_i) .

$$\rho_p = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)}\sqrt{\text{Var}(Y)}} = \frac{E(XY) - E(X)E(Y)}{\sqrt{\text{Var}(X)}\sqrt{\text{Var}(Y)}}$$

$$\hat{\rho}_p = \frac{\sum_i (X_i - \bar{X}_n)(Y_i - \bar{Y}_n)}{\sqrt{\sum_i (X_i - \bar{X}_n)^2} \sqrt{\sum_i (Y_i - \bar{Y}_n)^2}}$$

Hypothese:

a) $\rho_p = 0$ vs. $H_1 : \rho_p \neq 0$

b) $\rho_p \leq 0$ vs. $H_1 : \rho_p > 0$

c) $\rho_p \geq 0$ vs. $H_1 : \rho_p < 0$

Teststatistik:

$$T_n = \sqrt{n-2} \frac{\hat{\rho}_p}{\sqrt{1 - \hat{\rho}_p^2}} \sim t_{n-2}$$

Test-Verfahren:

Es seien $(x_i, y_i)_{1 \leq i \leq n}$ Ausprägungen von (X_i, Y_i) . Weiter sei

$$r_p = \frac{\sum_i (x_i - \bar{x}_n)(y_i - \bar{y}_n)}{\sqrt{\sum_i (x_i - \bar{x}_n)^2} \sqrt{\sum_i (y_i - \bar{y}_n)^2}}$$

$$t_n = \sqrt{n-2} \frac{r_p}{\sqrt{1 - r_p^2}}$$

Testentscheidung:

- a) $|t_n| > t_{n-2, 1-\alpha/2}$ H_0 verwerfen
- b) $t_n > t_{n-2, 1-\alpha}$ H_0 verwerfen
- c) $t_n < t_{n-2, \alpha}$ H_0 verwerfen

Rank-Korrelationstest

Voraussetzung:

$$(X_i, Y_i) \stackrel{i.i.d.}{\sim} F$$

wobei X_i und Y_i stetige oder ordinal-skalierte Merkmale sind.

Sei ρ_s der theoretische und $\hat{\rho}_s$ der empirische Spearman-Korrelationskoeffizient von (X_i, Y_i) .

Hypothese:

a) $\rho_s = 0$ vs. $H_1 : \rho_s \neq 0$

b) $\rho_s \leq 0$ vs. $H_1 : \rho_s > 0$

c) $\rho_s \geq 0$ vs. $H_1 : \rho_s < 0$

Rank-Korrelationstest, approximativ (2)

Teststatistik:

$$T_n = \sqrt{n-2} \frac{\hat{\rho}_s}{\sqrt{1 - \hat{\rho}_s^2}} \approx t_{n-2}$$

Test-Verfahren:

Es seien $(x_i, y_i)_{1 \leq i \leq n}$ Ausprägungen von (X_i, Y_i) . Weiter sei r_s der berechnete Wert für $\hat{\rho}_s$.

$$t_n = \sqrt{n-2} \frac{r_s}{\sqrt{1 - r_s^2}}$$

Testentscheidung:

- a) $|t_n| > t_{n-2, 1-\alpha/2}$ H_0 verwerfen
- b) $t_n > t_{n-2, 1-\alpha}$ H_0 verwerfen
- c) $t_n < t_{n-2, \alpha}$ H_0 verwerfen

Rank-Korrelationstest, exakt (3)

Teststatistik:

$$T_n^1 = \frac{(n-1)n(n+1)}{6}(1 - \hat{\rho}_s) \quad \text{exakte Verteilung ist bekannt.}$$

Test-Verfahren:

Es seien $(x_i, y_i)_{1 \leq i \leq n}$ Ausprägungen von (X_i, Y_i) . Weiter sei r_s der berechnete Wert für $\hat{\rho}_s$.

$$t_n^{(1)} = \frac{(n-1)n(n+1)}{6}(1 - r_s)$$

Testentscheidung:

Der Wert von $t_n^{(1)}$ wird mit dem Quantil der Verteilung von T_n^1 verglichen.

cor.test()

Beschreibung

Test auf Zusammenhang zwischen paarweisen Stichproben mittels Pearsons Korrelationskoeffizient, Kendalls tau oder Spearmans rho.

Anwendung

```
cor.test(x, y, alternative = c("two.sided", "less", "greater"),  
method = c("pearson", "kendall", "spearman"),  
exact = NULL, conf.level = 0.95, continuity = FALSE)
```

```
cor.test(formula, data, subset, na.action)
```

Argumente

x , *y* numerische Vektoren von Datenwerten. *x* und *y* müssen die gleiche Länge haben.

exact logischer Wert, gibt an ob der exakte p-Wert berechnet werden soll. Nur bei Verwendung von Kendalls tau oder Spearmans rho.

continuity logischer Wert: bei TRUE wird eine Stetigkeitskorrektur durchgeführt, falls Kendalls tau oder Spearmans rho ohne Berechnung des exakten p-Werts benutzt wird.

5. Statistische Tests

- 1 Binomialtest
- 2 t-Test (Mittelwertvergleich)
- 3 Multiple Test
- 4 F-Test (Varianzvergleich)
- 5 Anpassungstests
- 6 Testen auf Unabhängigkeit
- 7 Varianzanalyse (Mittelwertvergleich $n > 2$)
- 8 Regressionsanalyse

Einfaktorielle Varianzanalyse (One Way ANOVA): Beispiel

In der PISA-Studie 2001 wurde u.a. das (qualitative) Merkmal *Zeitaufwand der Schüler für Hausaufgaben* mit den Ausprägungen gering(1), mittel(2), groß(3) und das (quantitative) Merkmal *mathematische Grundbildung der Schüler* (Ausprägungen: erreichte Punktzahl) erfasst.

Fragestellung: unterscheidet sich die Verteilung des Merkmals *mathematische Grundbildung* in den 3 Gruppen?

Einfaktorielle Varianzanalyse

Gruppe 1		Gruppe 2		Gruppe 3	
Land	Punkte	Land	Punkte	Land	Punkte
FIN	536	AUS	533	GR	447
J	557	B	520	GB	529
FL	514	BR	334	IRL	503
L	446	DK	514	I	457
A	515	D	490	LV	463
S	510	F	517	MEX	387
CH	529	IS	514	PL	470
CZ	498	CDN	533	RUS	478
		ROK	547	E	476
		NZ	537	H	488
		N	499		
		P	454		
		USA	493		
\bar{x}_1	513.1	\bar{x}_2	498.8	\bar{x}_3	469.8

Einfaktorielle Varianzanalyse (2)

Annahmen/Schreibweise:

x_{ij} : Realisierung der ZV X_{ij} , für $i = 1, \dots, I$, $j = 1, \dots, n_i$
 j -te Beobachtung aus der i -ten Gruppe

$$X_{ij} \sim \mathcal{N}(\mu_i, \sigma^2)$$

Varianz muss identisch sein: σ^2

$$\sum_{i=1}^I n_i = n$$

H_0 vs. H_1 :

$$H_0 : \mu_1 = \mu_2 = \dots = \mu_I$$

$$H_1 : \exists i_1, i_2 \ (i_1 \neq i_2), \text{ mit: } \mu_{i_1} \neq \mu_{i_2}$$

Einfaktorielle Varianzanalyse (3)

Test-Verfahren:

1. Berechne den gesamten Mittelwert:

$$\bar{x} = \frac{1}{n} \sum_{i,j} x_{ij} = \frac{1}{n} \sum_{i=1}^I \sum_{j=1}^{n_i} x_{ij}$$

2. Berechne die Mittelwerte der einzelnen Gruppen:

$$\bar{x}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{ij}$$

3. Berechne die gewichtete Streuung der Mittelwerte \bar{x}_i um den Gesamtmittelwert \bar{x} (*Streuung zwischen den Gruppen*):

$$SSB = \sum_{i=1}^I n_i (\bar{x}_i - \bar{x})^2 \quad (\text{Sum of Squares, between}) \quad (5.17)$$

Einfaktorielle Varianzanalyse (4)

```
d1 <- c(536, 557, 514, 446, 515, 510, 529, 498)
d2 <- c(533, 520, 334, 514, 490, 517, 514, 533, 547, 537,
        499, 454, 493)
d3 <- c(447, 529, 503, 457, 463, 387, 470, 478, 476, 488)
x1 <- mean(d1); x2<-mean(d2); x3<-mean(d3); x1;x2;x3
x <- mean(c(d1,d2,d3)); x
ssb <- 8*(x1-x)**2+13*(x2-x)**2+10*(x3-x)**2
ssb
```

9066.026

Einfaktorielle Varianzanalyse (5)

Test-Verfahren:

Große Gruppen sollen stärkeres Gewicht haben als kleine Gruppen.

4. Man kann zeigen, dass

$$\frac{1}{I-1}SSB = \frac{1}{I-1} \sum_{i=1}^I n_i (\bar{x}_i - \bar{x})^2 \sim \chi_{I-1}^2$$

Folglich kann man den Wert $\frac{1}{I-1}SSB$ mit dem Quantil der χ^2 -Verteilung vergleichen.

Das folgende Beispiel zeigt, dass die Größe $\frac{1}{I-1}SSB$ allein keine geeignete Teststatistik zur Überprüfung der H_0 ist.

data1:

	Gr1	Gr2	Gr3
1	50	48	57
2	42	57	59
3	53	65	48
4	45	59	46
5	55	51	45
\bar{x}_i	49	56	52

$$\bar{x}=52$$

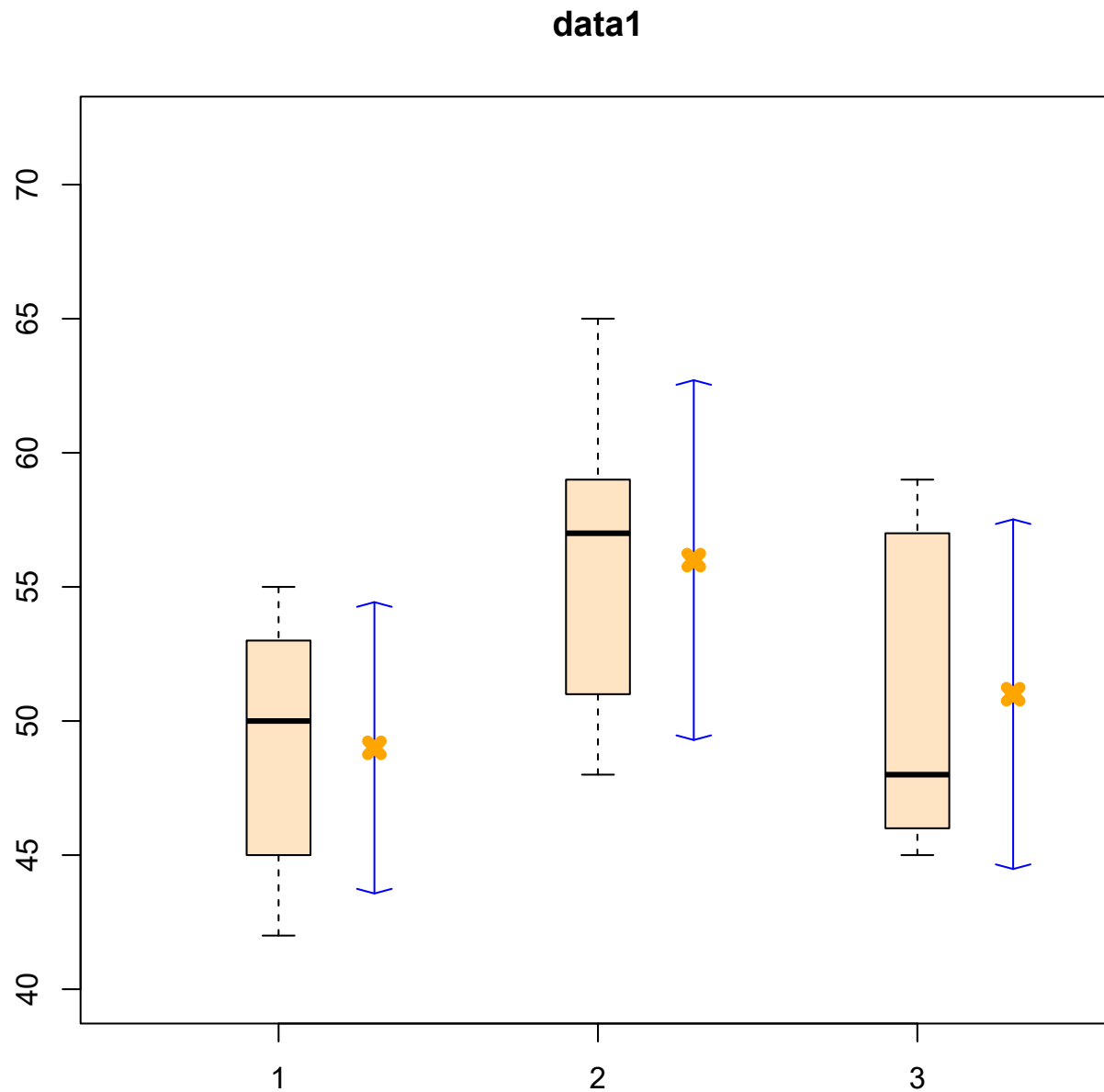
data2:

	Gr1	Gr2	Gr3
1	47	55	53
2	53	54	50
3	49	58	51
4	50	61	52
5	46	52	49
\bar{y}_i	49	56	52

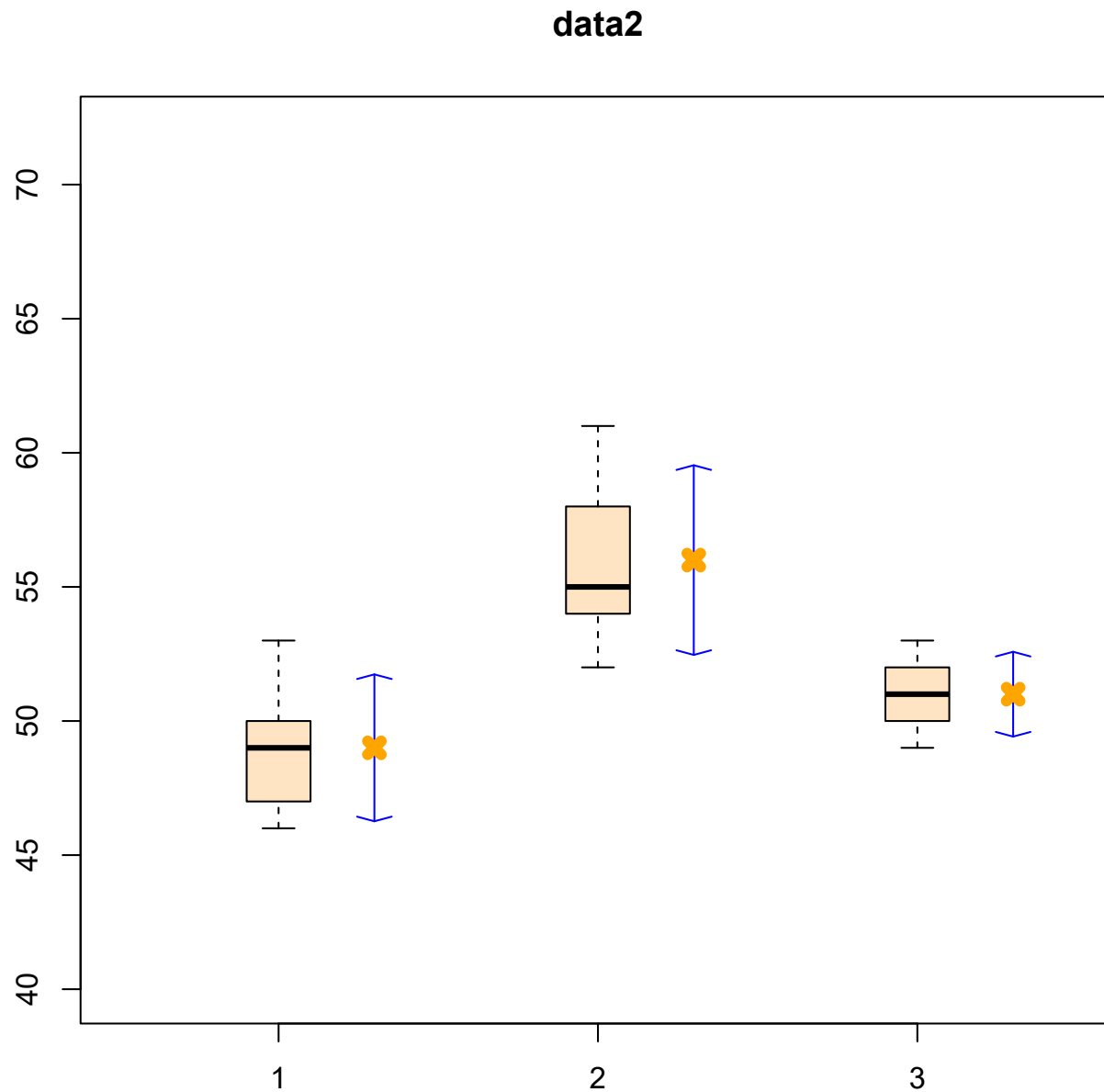
$$\bar{y}=52$$

Folglich ist der Wert von $\frac{1}{I-1}SSB$ in beiden Datensätzen identisch.

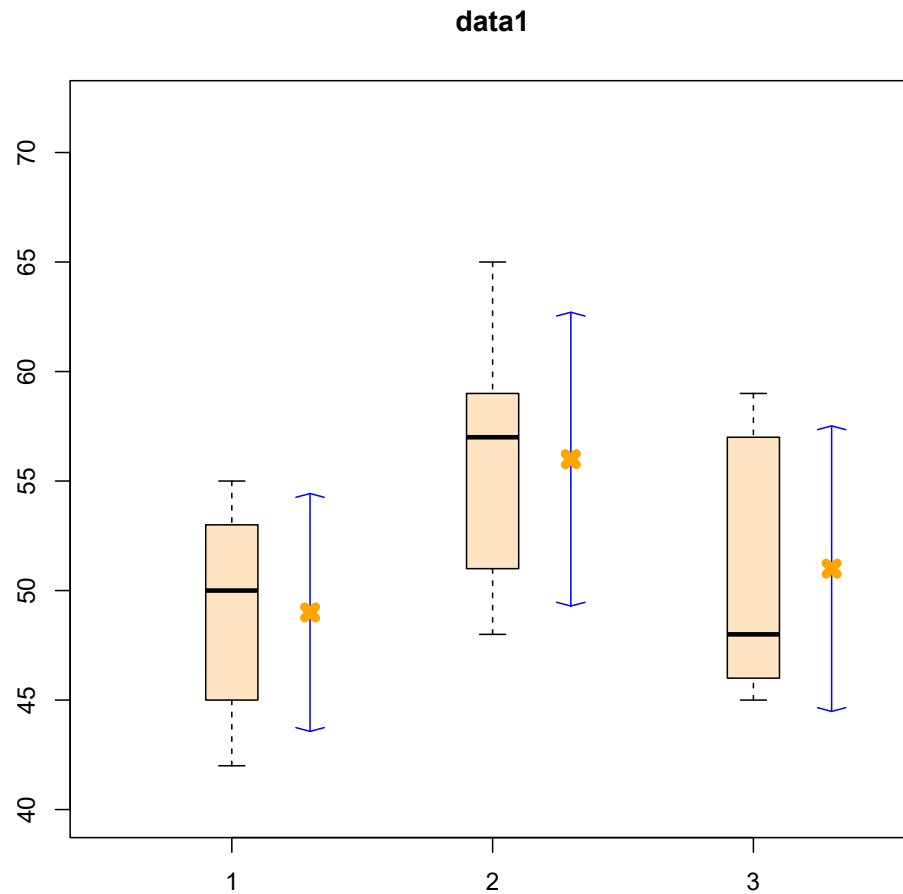
Betrachten wir die Verteilung der Daten in den 3 Gruppen der beiden Datensätze genauer...



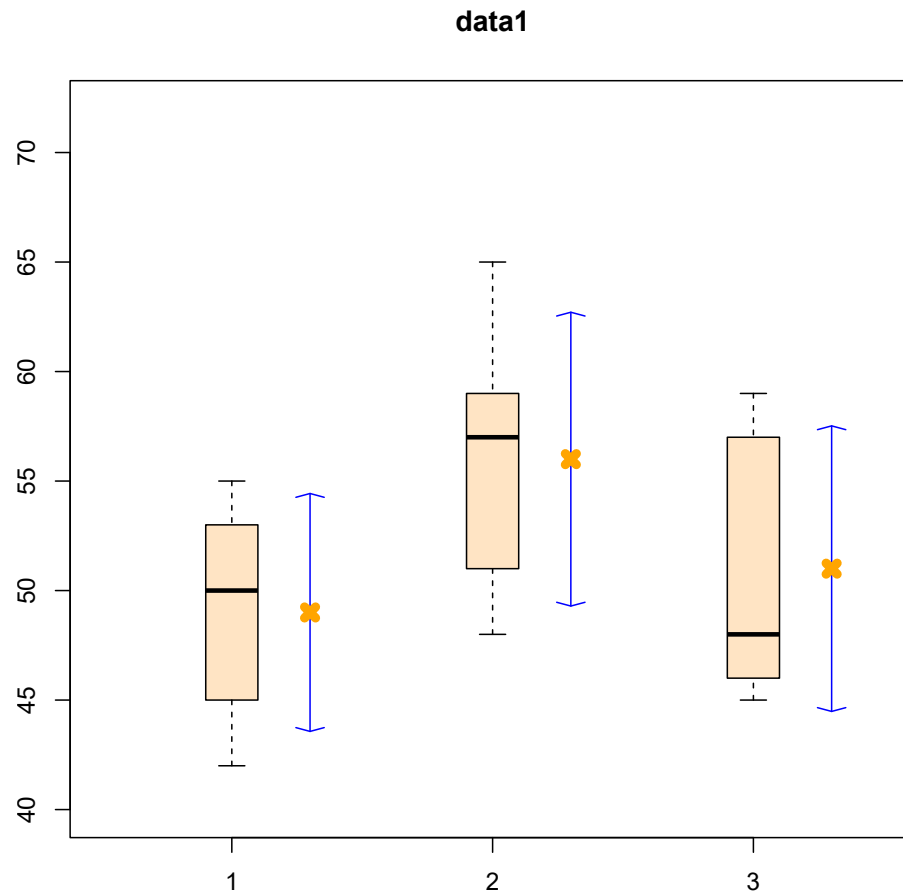
Oranges x und blaue Linie: $\bar{x}_i \pm sd(x_i)$: Mittelwert und Standardabw. in der i -ten Gruppe



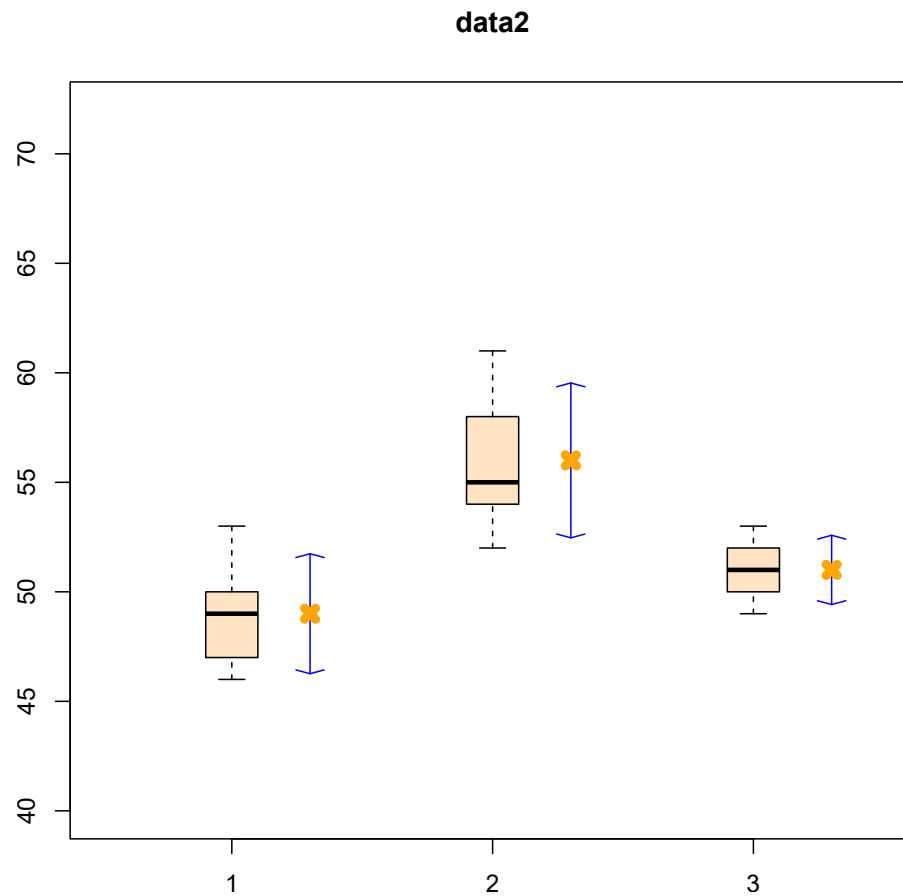
Oranges \bar{x} und blaue Linie: $\bar{x}_i \pm sd(x_i)$: Mittelwert und Standardabw. in der i -ten Gruppe



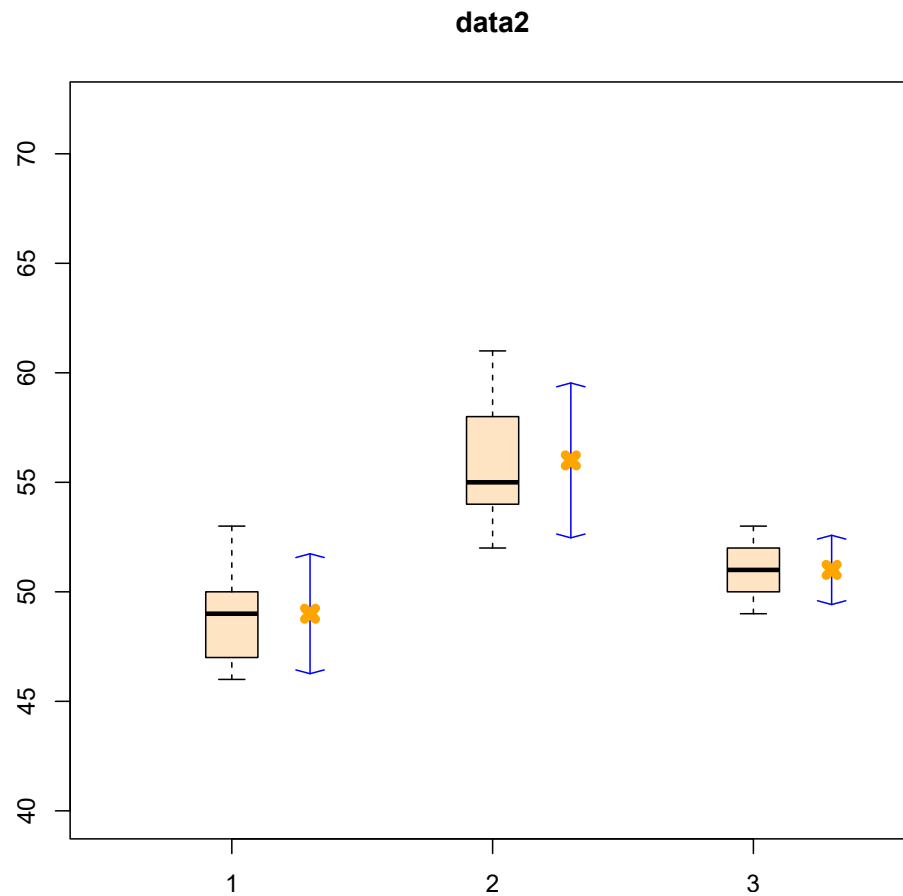
- ▶ **Streuung innerhalb der Gruppen ist groß.**
- ▶ Unterschiedliche Mittelwerte können durch die hohen Streuungen erklärt werden.



- ▶ Streuung innerhalb der Gruppen ist groß.
- ▶ Unterschiedliche Mittelwerte können durch die hohen Streuungen erklärt werden.



- ▶ Streuung innerhalb der Gruppen ist klein.
- ▶ Spricht für einen Lageunterschied zwischen den Gruppen.



- ▶ Streuung innerhalb der Gruppen ist klein.
- ▶ Spricht für einen Lageunterschied zwischen den Gruppen.

Einfaktorielle Varianzanalyse (6)

Man muss neben der Streuung zwischen den Gruppen die Streuung innerhalb der Gruppen berücksichtigen:

Für data1: $s_1^2 = 29.5$, $s_2^2 = 45.0$, $s_3^2 = 42.5$

Für data2: $s_1^2 = 7.5$, $s_2^2 = 12.5$, $s_3^2 = 2.5$

Test-Verfahren:

5. Man berechnet die Streuung innerhalb jeder Gruppe (SSW_i)

$$SSW_i = \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)^2$$

und summiert über alle Gruppen:

$$SSW = \sum_{i=1}^I SSW_i = \sum_{i=1}^I \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)^2 \text{ (Sum of Squares, within)}$$

Einfaktorielle Varianzanalyse (7)

Test-Verfahren:

6. Die Gesamtstreuung kann man berechnen durch

$$SST = SSB + SSW = \sum_{i=1}^I n_i (\bar{x}_i - \bar{x})^2 + \sum_{i=1}^I \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)^2 \quad (5.18)$$

oder äquivalent:

$$SST = \sum_{i=1}^I \sum_{j=1}^{n_i} (x_{ij} - \bar{x})^2 \quad (\text{Sum of Squares, total}) \quad (5.19)$$

Einfaktorielle Varianzanalyse (8)

Test-Verfahren:

7. Man berechnet die mittlere Streuung zwischen den Gruppen (MSSB) und die mittlere Streuung innerhalb der Gruppen (MSSW) und betrachtet ihr Verhältnis:

$$F = \frac{MSSB}{MSSW} = \frac{\frac{1}{I-1} \sum_{i=1}^I n_i (\bar{X}_i - \bar{X})^2}{\frac{1}{n-I} \sum_{i=1}^I \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_i)^2} \quad (5.20)$$

Einfaktorielle Varianzanalyse (9)

Test-Verfahren:

Aus:

$$MSSB = \frac{1}{I-1}SSB \sim \chi_{I-1}^2 \quad \text{und} \quad MSSW = \frac{1}{n-I}SSW \sim \chi_{n-I}^2$$

folgt:

$$F = \frac{MSSB}{MSSW} \sim F_{I-1, n-I} \quad (5.21)$$

8. Ist $F > F_{I-1, n-I; 1-\alpha}$, dann wird H_0 verworfen.

$F_{I-1, n-I; 1-\alpha}$: $1 - \alpha$ -Quantil der F-Verteilung mit $I - 1$ und $n - I$ Freiheitsgraden.

Einfaktorielle ANOVA (10): Bsp. PISA-Studie

```
d1 <- c(536, 557, 514, 446, 515, 510, 529, 498)
d2 <- c(533, 520, 334, 514, 490, 517, 514, 533, 547, 537,
        499, 454, 493)
d3 <- c(447, 529, 503, 457, 463, 387, 470, 478, 476, 488)
x1 <- mean(d1); x2<-mean(d2); x3<-mean(d3);
x <- mean(c(d1,d2,d3))
ssb <- 8*(x1-x)**2+13*(x2-x)**2+10*(x3-x)**2
mssb <- ssb/2
n <- length(d1)+length(d2)+length(d3)
ssw1 <- sum((d1-mean(d1))**2)
ssw2 <- sum((d2-mean(d2))**2)
ssw3 <- sum((d3-mean(d3))**2)
ssw <- ssw1+ssw2+ssw3
mssw <- ssw/(n-3)
```

Einfaktorielle ANOVA (11): Bsp. PISA-Studie

```
f.stat <- mssb/mssw  
cv <- qf(0.95, df1=2, df2=n-3)  
f.stat  
2.238  
  
cv  
3.34  
  
# Pr(>F)  
1-pf(f.stat, df1=2, df2=n-3)  
0.125
```

$f.stat = 2.238 \leq cv = 3.34 \Rightarrow H_0$ wird nicht abgelehnt!

Einfaktorielle ANOVA (12)

ANOVA-Tabelle:

Source of Variation	df	SS	MS	F	$P(f > F)$
Between	$I-1$	SS_B	MSS_B	$\frac{MMS_B}{MSS_W}$	
Within	$n-I$	SS_W	MSS_W		
Total	$n-1$	SS_T			

Für unser Bsp.:

Source of Variation	df	SS	MS	F	$P(f > F)$
Between	2	9066	4533	2.237	0.1254
Within	28	56720	2026		
Total	30	65786			

```
data1 <- data.frame(  
  x=c(d1,d2,d3),  
  g=as.factor(rep(c(1,2,3),c(8,13,10)))  
) # oder mit stack()  
head(data1)  
summary(aov(formula=x~g, data=data1))  
# oder:  
anova(lm(formula=x~g, data=data1))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
g	2	9066	4533.0	2.2377	0.1254
Residuals	28	56720	2025.7		

aov(), oneway.test()

Beschreibung

Passe ein ANOVA (analysis of variance)-Modell an mittels Aufruf von `lm`.

Usage

aov(formula, data = NULL, ...)

Arguments

formula Formel, die das Modell spezifiziert.

data Data frame, in dem sich die Variablen aus der Formel befinden. Bei Fehlen werden die Variablen wie gewöhnlich gesucht (im Speicher).

Siehe auch:

oneway.test(formula, data, subset, var.equal = FALSE) insbesondere wenn keine Varianzhomogenität vorliegt.

Übung 5.12

In einer Studie wurde der Effekt unterschiedlicher Nährlösungen (Zucker) auf das Wachstum von Erbsen untersucht. Die Ergebnisse sind in *erbsen.txt* in *Stud.IP* zu finden. Die Daten enthalten die Länge der Erbsen (*length*) und die Behandlungsgruppen (*treat*: c (Kontrolle), 2g (2% Glukose), 2f (2% Fruktose), 1g1f (1% Glukose + 1% Fruktose) und 2s (2% Saccharose)). Testen Sie mit Hilfe der Varianzanalyse, ob sich die Mittelwerte der Erbsen-Längen in den 5 Gruppen unterscheiden.

Verwenden Sie das Signifikanzniveau $\alpha = 0.05$.

Lösung

```
erbsen <- read.csv("erbsen.txt", sep=";")
boxplot(formula=length~treat, data=erbsen)
aov1 <- aov(formula=length~treat, data=erbsen)
summary(aov1)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
treat	4	1077.3	269.33	82.17	<2e-16 ***
Residuals	45	147.5	3.28		

—
Signif. codes: 0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Ablehnung der Nullhypothese, dass alle Mittelwerte gleich sind. Der Test zeigt nicht, welche Gruppen sich signifikant voneinander unterscheiden. Dafür benutzt man z.B. einen paarweisen t -Test (mit Korrektur für multiples Testen)

```
pairwise.t.test(erbsen$length, erbsen$treat, p.adj="holm")
```

oder Tukey's HSD (honestly significant difference)-Test:

```
TukeyHSD(aov1)
```

Zweifache Varianzanalyse (Two Way ANOVA)

Beispiel:

Es wird der Einfluss zweier Faktoren (unabhängige Variablen) auf die Depressivität (abhängige Variable) untersucht. Bsp: Wirksamkeit einer Behandlungsform (Plazebo, geringe Dosierung, mehr Dosierung) unter Berücksichtigung des Geschlechtes (M, F) auf eine Variable:

I) $Y_{ijk} = \alpha_i + \beta_j + \varepsilon_{ijk}$ ohne Wechselwirkung

II) $Y_{ijk} = \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ijk}$ mit Wechselwirkung

I) M1 `<-lm(y ~x1 + x2)`

II) M2 `<-lm(y ~x1 *x2)`

Zweifache Varianzanalyse: Beispiel

Geschlecht	Medikament	Plazebo	Gering	Hoch
M		22	16	13
M		25	16	12
M		22	16	12
M		21	15	13
M		22	15	12
F		18	19	16
F		19	20	14
F		17	17	16
F		21	16	13
F		19	16	14

Zweifache Varianzanalyse: Beispiel

```
y <- c (22 , 25 , 22 , 21 , 22 , 16 , 16 , 16 , 15 , 15 ,  
        13 , 12 , 12 , 13 , 12 , 18 , 19 , 17 , 21 , 19 ,  
        19 , 20 , 17 , 16 , 16 , 16 , 14 , 16 , 13 , 14)  
sex <- as.factor(c(rep("M",15),rep("F",15)))  
beh <- as.factor(c(rep("Plazebo",5),  
                    rep("Gering",5),rep("Hoch",5)))  
dat <- data.frame(y, sex, beh)  
M1 <- lm(y~sex*beh, data=dat)  
library(car)  
Anova(M1)
```

Signifikanter Haupteffekt der Behandlung und signifikante Wechselwirkung, aber kein signifikanter Haupteffekt des Geschlechtes.

```
interaction.plot(dat$beh, dat$sex, dat$y)
```

ANOVA type I, II oder III

Daten balanciert: kein Unterschied zwischen type I, II, III. Man kann hier die Funktion `anova()` (=type I) benutzen:

```
anova(M1)
```

Daten unbalanciert: ANOVA vom type II oder III. Beispiel mit der Funktion `Anova()` aus dem Package `car` für type II:

```
library(car)  
Anova(M1)
```

Daten unbalanciert mit Interaktionseffekt: type III notwendig; dazu unbedingt orthogonale Kontraste für das lin. Modell verwenden:

```
M2 <- lm(y~sex*beh, data=dat,  
         contrasts=list(sex=contr.sum, beh=contr.sum))  
Anova(M2, type=3)
```

Statt die Kontraste an `lm()` zu übergeben kann man den Default ändern (bevor man `lm()` bzw `aov()` ausführt):

```
options(contrasts = c("contr.sum", "contr.poly"))  
# wenn man wieder Standard-Kontraste haben will:  
# options(contrasts = c("contr.treatment", "contr.poly" ))
```

5. Statistische Tests

- 1 Binomialtest
- 2 t-Test (Mittelwertvergleich)
- 3 Multiple Test
- 4 F-Test (Varianzvergleich)
- 5 Anpassungstests
- 6 Testen auf Unabhängigkeit
- 7 Varianzanalyse (Mittelwertvergleich $n > 2$)
- 8 Varianzanalyse und Regressionsanalyse

Varianzanalyse und lineare Regressionsanalyse

$$Y \sim X$$

- Y abhängige Variable, bzw. Regressand, Response.
- X unabhängige Variable, bzw. Regressor, erklärende Variable.

einfache lineare Regressionsanalyse:

Y ist numerisch,
 X ist numerisch.

einfaktorielle Varianzanalyse:

Y ist numerisch,
 X kategorial.

Varianzanalyse und lineare Regressionsanalyse

Einfache lineare Regressionsanalyse:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \quad \varepsilon_i \stackrel{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$

Y_i : abhängige Variable, Regressand, Response

X_i : unabhängige Variable, Regressor, erklärende Variable

Einfaktorielle Varianzanalyse:

$$Y_{ij} = \mu_j + \varepsilon_{ij}, \quad \varepsilon_{ij} \stackrel{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$

Y_{ij} : abhängige Variable, Regressand, Response

μ_j : erklärende Variable

Varianzanalyse lässt sich als ein Spezialfall der multiplen Regressionsanalyse mit $I - 1$ Dummy-Variablen darstellen.

Varianzanalyse und lineare Regressionsanalyse

Varianzanalyse als Spezialfall der multiplen Regressionsanalyse:

$$Y_{ij} = \mu_i + \varepsilon_{ij} \Leftrightarrow$$

$$Y_{ij} = \beta_0 + \beta_1 X_{2,j} + \beta_2 X_{3,j} + \dots \beta_{I-1} X_{I,j} + \varepsilon_{ij}$$

wobei:

$$X_{2,j} = \mathbb{1}_{\{i=2\}}, \dots, X_{I,j} = \mathbb{1}_{\{i=I\}}$$

mit:

$$\beta_0 = \mu_1, \quad \beta_1 = \mu_2 - \mu_1, \dots \quad \beta_{I-1} = \mu_I - \mu_1$$

Einfache lineare Regressionsanalyse

Beispiel:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Vorausgesetzt:

- $E(\varepsilon_i) = 0$
- $Var(\varepsilon_i) = \sigma^2$
- $Cov(\varepsilon_i, \varepsilon_j) = 0, \quad \forall i \neq j$

Bemerkung: nicht vorausgesetzt: $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$

Mit Hilfe von Daten (x_i, y_i) sollen die Parameter β_0, β_1 und σ geschätzt werden.

Methode der kleinsten Quadrate (Least Square Method: LS Method)

Einfache lineare Regressionsanalyse

LS Methode:

- *fitted values*: $\hat{Y}_i = \beta_0 + \beta_1 X_i$
- Residuen (Schätzfehler): $E_i = Y_i - \hat{Y}_i$
- Minimiere *Residual Sum of Squares* RSS:

$$RSS(\beta_0, \beta_1) = \sum_{i=1}^n E_i^2 = \sum_{i=1}^n (Y_i - \beta_0 - \beta_1 X_i)^2$$

$$\frac{\partial RSS(\beta_0, \beta_1)}{\partial \beta_0} = 0$$

$$\frac{\partial RSS(\beta_0, \beta_1)}{\partial \beta_1} = 0$$

Einfache lineare Regressionsanalyse

⇒ LS-Schätzer (*LS estimator-LSE*):

$$\hat{\beta}_1 = \frac{S_{XY}}{S_{XX}} = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2}, \quad \hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X} \quad (5.22)$$

σ Schätzen?

$$\sigma^2 = \text{Var}(\varepsilon_i)$$

- *fitted Model*: $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$

- berechne Residuen $E_i = Y_i - \hat{Y}_i = Y_i - (\hat{\beta}_0 + \hat{\beta}_1 X_i)$

- berechne $RSS(\hat{\beta}_0, \hat{\beta}_1) = \sum_{i=1}^n E_i^2$

$$\hat{\sigma}^2 = \frac{RSS}{n-2}, \quad (\text{wegen 2 geschätzten Parametern } \beta_0 \text{ und } \beta_1) \quad (5.23)$$

Streuungszerlegung

$$TSS = \sum_{i=1}^n (Y_i - \bar{Y})^2 \quad \text{Total sum of squares}$$

$$RSS = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad \text{Residual sum of squares}$$

$$RegSS = \sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2 \quad \text{Regression sum of squares}$$

Man kann zeigen:

$$TSS = RegSS + RSS$$

R^2 : das Bestimmtheitsmaß

$$TSS = RegSS + RSS$$

Totale Variabilität = durch Regression erklärte Variabilität + Rest

$$\sum_{i=1}^n (Y_i - \bar{Y})^2 = \sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2 + \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

$$R^2 = \frac{RegSS}{TSS} = 1 - \frac{RSS}{TSS} \quad , \quad 0 \leq R^2 \leq 1 \quad (5.24)$$

R^2 gibt den Teil der totalen Variation an, der durch das angepasste Regressionmodell erklärt wird.

Einfache lineare Regressionsanalyse

Warum setzt man zusätzlich $\varepsilon_i \stackrel{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$ voraus?

Mit dem Modell $Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$ und Hilfe der LS-Methode haben wir $\hat{\beta}_i$ geschätzt.

Unterstellt man den Fehlern ε_i eine Normalverteilung, dann kann man testen, **ob die geschätzten Parameter signifikant von 0 verschieden sind.**

Dies hilft abzuklären, ob der geschätzte Einfluß von X auf Y ein statistisch valider Zusammenhang ist, oder nur ein numerisches Artefakt ist.

Einfache lineare Regressionsanalyse

Um die Hypothesentests durchführen zu können ($H_1 : \beta_i \neq 0$) und Konfidenzintervalle für die Schätzer $\hat{\beta}_i$ aufzustellen, müssen wir ihre Verteilungen kennen.

Man unterstellt z.B., dass $\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \Rightarrow$

$$\hat{\beta}_i \sim \mathcal{N}(\beta_i, \text{Var}(\beta_i))$$

$\text{Var}(\beta_i)$ wird geschätzt durch:

$$\text{Var}(\hat{\beta}_0) = \frac{\hat{\sigma}^2 \sum_{i=1}^n X_i^2}{nS_{XX}}$$

$$\text{Var}(\hat{\beta}_1) = \frac{\hat{\sigma}^2}{S_{XX}}, \quad \text{mit} \quad S_{XX} = \sum_{i=1}^n (X_i - \bar{X})^2$$

Einfache lineare Regressionsanalyse

Verteilung der LS-Schätzer unter Normalverteilungsannahme:

$$\hat{\beta}_i \sim \mathcal{N}(\beta_i, \text{Var}(\beta_i)) \Rightarrow$$

$$\frac{\hat{\beta}_i - \beta_i}{\sqrt{\text{Var}(\hat{\beta}_i)}} \sim t_{n-2}$$

Somit kann man die Konfidenzintervalle für die Schätzer $\hat{\beta}_i$ bestimmen:

$$\hat{\beta}_i \pm \sqrt{\text{Var}(\hat{\beta}_i)} t_{n-2, 1-\alpha/2}$$

Einfache lineare Regressionsanalyse

Testen der Nullhypothese: $H_0 : \beta_1 = 0$:

1) Man verwendet den t-Test:

$$T = \frac{\hat{\beta}_1 - 0}{\sqrt{\text{Var}(\hat{\beta}_1)}} \sim t_{n-2}$$

2) oder den F-Test:

$$T^2 = \frac{(\hat{\beta}_1)^2}{\text{Var}(\hat{\beta}_1)} \sim F_{1,n-2}$$

Regressionsanalyse mit R: *lm()*

Beschreibung

`lm` wird benutzt um lineare Modelle anzupassen. Diese können benutzt werden um eine Regression durchzuführen, für einschichtige Varianzanalyse und Kovarianzanalyse.

Benutzung

`lm(formula, data, subset, weights, na.action, ...)`

Argumente

`formula` Formel, die das Modell spezifiziert. Details für die Modellspezifikation findet man in der Hilfe unter "Details".

`data` Ein Data frame, in dem die Variablen aus der Formel spezifiziert sind.

Beispiel

```
data(hills, package="MASS"); head(hills); ?hills
```

The record times in 1984 for 35 Scottish hill races.

dist: distance in miles,

climb: total height gained, in feet,

time: record time in minutes.

Modell 1: time~dist:

```
modell1 <- lm(time~dist, data=hills); modell1
```

```
lm(formula = time~dist, data = hills)
```

Coefficients:

(Intercept) dist

-4.841 8.330

Beispiel

Der Output für die Funktion *lm* (hier `model1`) enthält weitere nützliche Ergebnisse, die mit Hilfe anderer Funktionen, angewendet auf das Modell *model1*, dargestellt werden:

`summary`(`model1`) : Parameterschätzer und overall model fit
`plot`(`model1`) : Plots der Residuen, q-q, leverage
`coef`(`model1`) : Model-Koeffizienten β_i
`confint`(`model1`) : Konfidenzintervalle für Parameter
`resid`(`model1`) : Residuen e_i
`fitted`(`model1`) : vorhergesagte Werte \hat{y}_i
`abline`(`model1`) : fügt einem Scatter-Plot eine einfache lineare
Regressionslinie hinzu
`predict`(`model1`, `newdata=x1`) : vorhergesagte Werte für `x1`
`anova`(`model1`) : ANOVA-Tabelle
`anova`(`model1`, `model2`) : vergleiche Fits beider Modelle:
"full" vs "reduced"

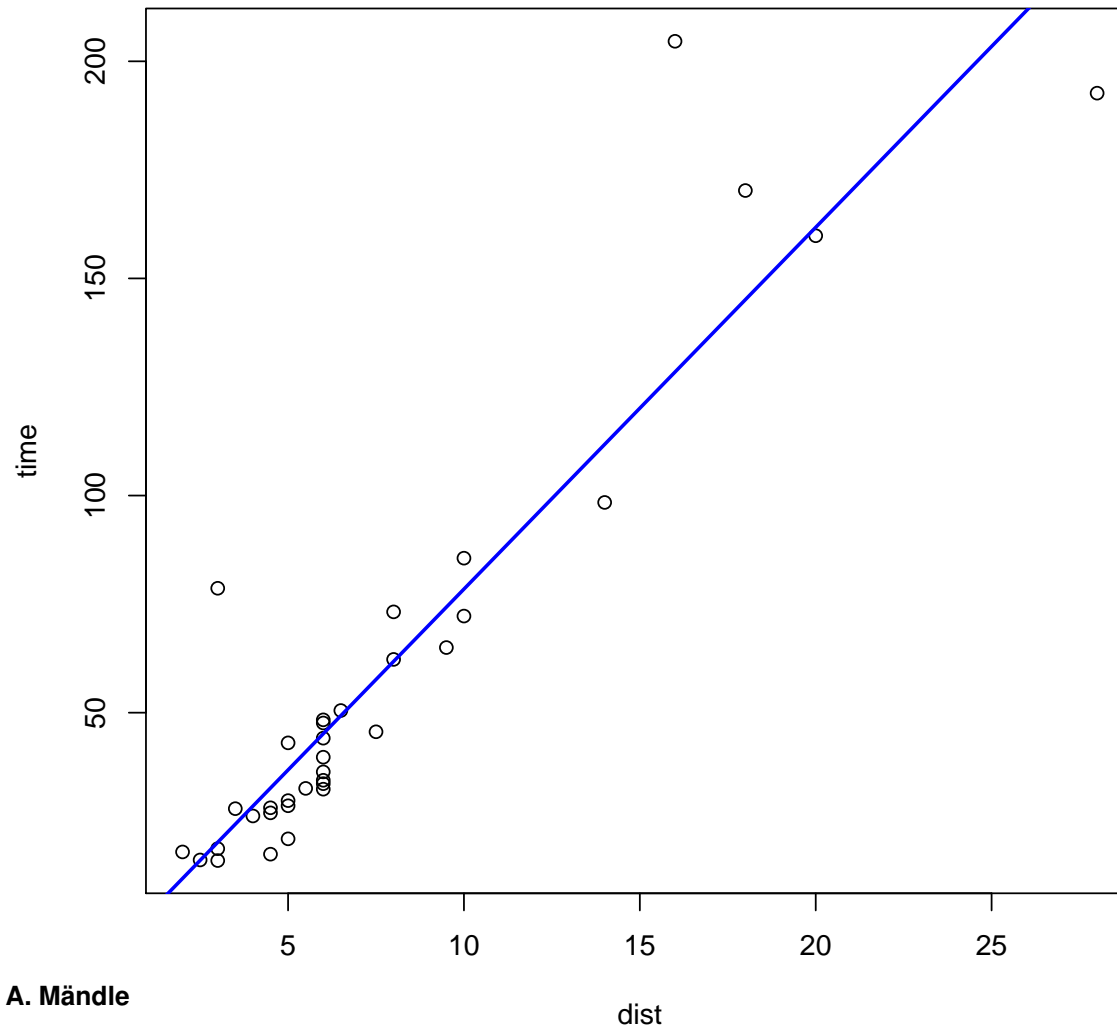
Beispiel: *summary()* und *anova()*

Lineares Modell anpassen, Summary (Parameterschätzer und Model fit) und ANOVA-Tabelle ausgeben:

```
modell <- lm(time~dist,data=hills)
summary(modell)
anova(modell)
```

Beispiel: *abline()*

```
attach(hills)
plot(dist, time, xlab="distance_mile", ylab="time_minutes")
abline(model1, lwd = 2, col = "blue")
```



Beispiel: Modell ohne Achsenabschnitt

intercept gleich Null setzen ergibt für das Modell:

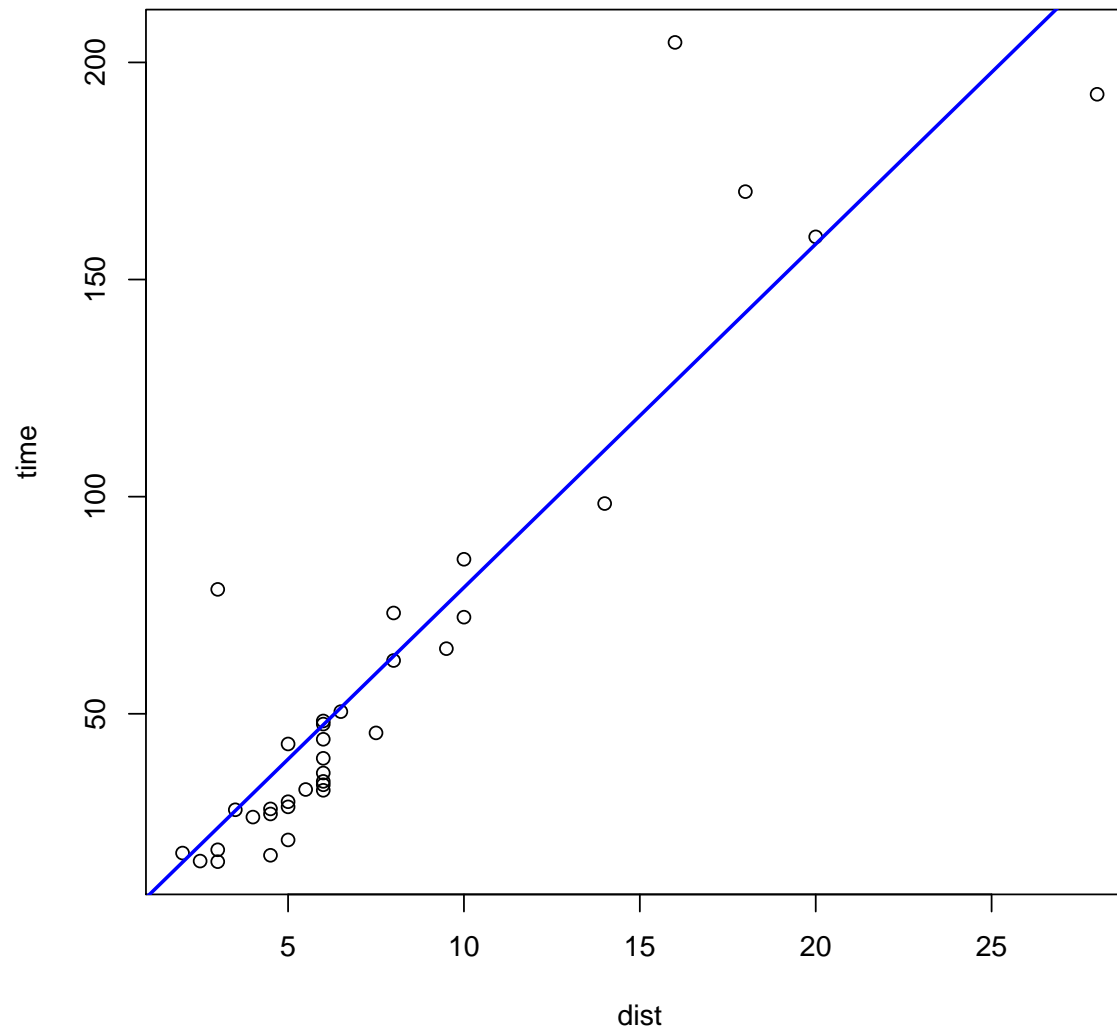
$$\beta_0 = 0 \Rightarrow Y_i = \beta_1 X_i + \varepsilon_i$$

in :

```
modell1 <- lm(time~dist-1, data=hills)
# oder
modell1 <- lm(time~dist+0, data=hills)
summary(modell1)
anova(modell1)
```


Beispiel: Modell ohne Achsenabschnitt

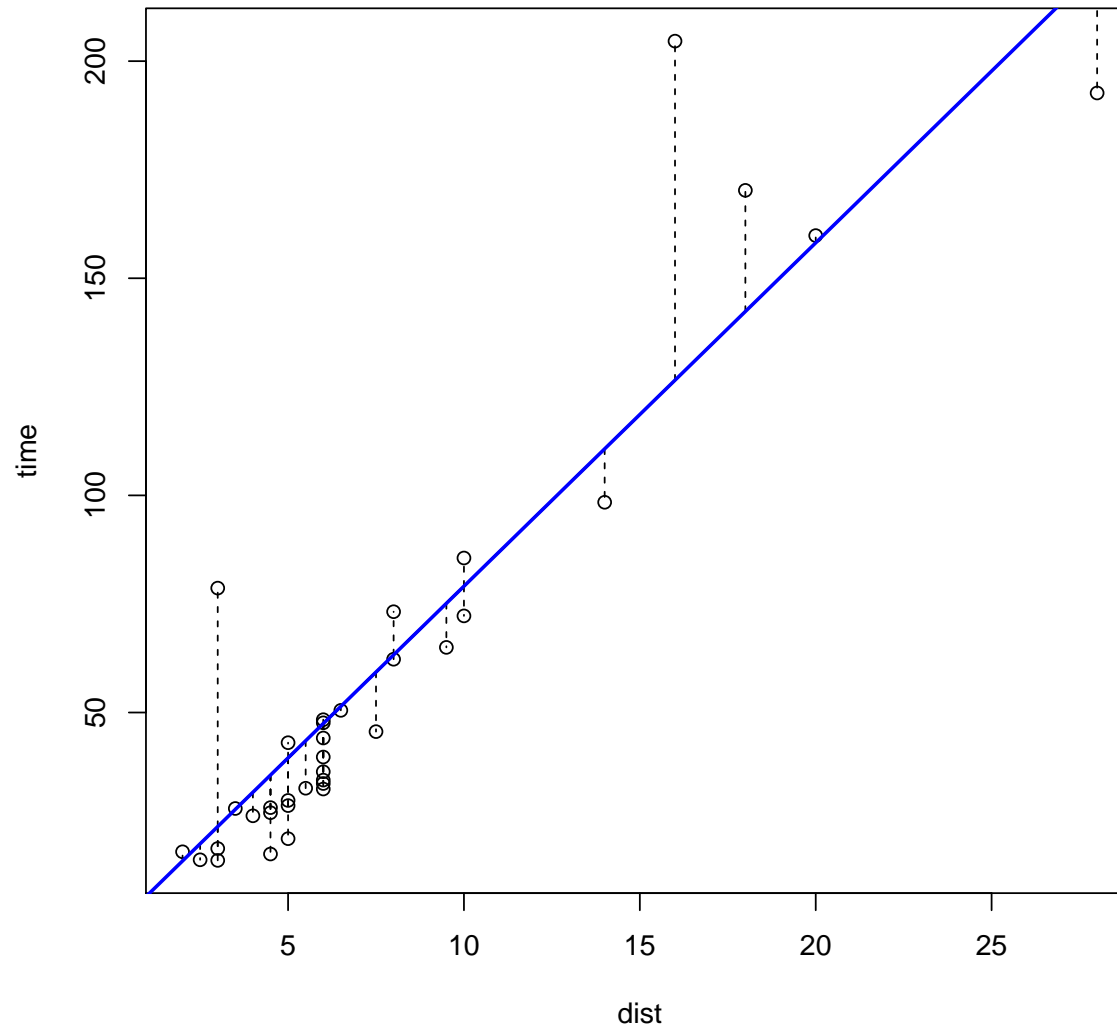
```
plot(dist, time, xlab="distance_mile", ylab="time_minutes")  
abline(model1, lwd = 2, col = "blue")
```



Beispiel: *fitted()*

```
title("Residual_plot")  
yhat <- fitted(model1)  
# lines(dist,yhat) # Werte identisch mit abline()  
segments(dist,yhat,dist,time,lty=2)
```

Residual plot

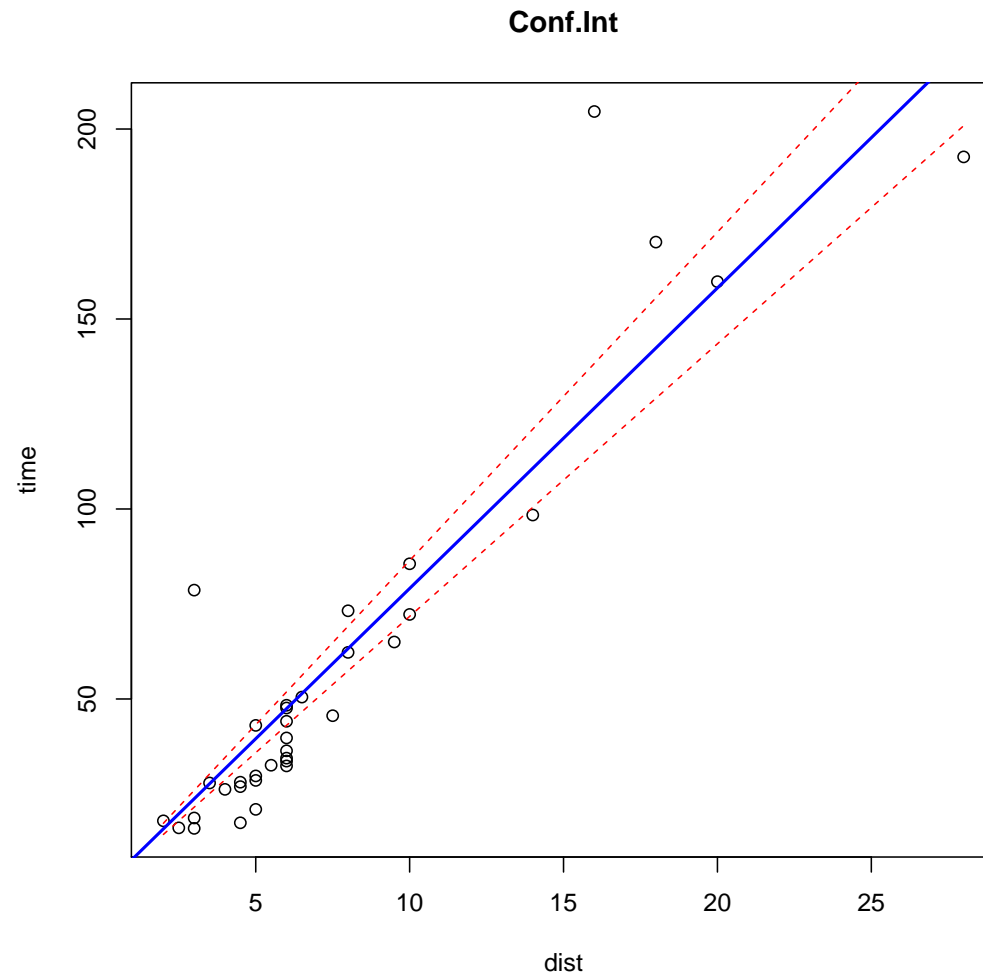


Beispiel: *predict()*

Plot der Vorhersagen mit Konfidenzintervall:

```
plot(dist, time, main="Conf.Int",  
      xlab="distance_mile", ylab="time_minutes")  
abline(modell1, lwd = 2, col = "blue")  
d <- seq(from = min(dist), to = max(dist), by = 0.01)  
data1 <- data.frame(dist = d)  
pre <- predict(modell1, newdata = data1,  
               interval = "confidence", level=0.95)  
lines(d, pre[,2], lty = 2, col = "red")  
lines(d, pre[,3], lty = 2, col = "red")
```

Beispiel: *predict()*



Beispiel: *plot.lm()*, *resid()*, *confint()*

Annahmen grafisch überprüfen:

```
?plot.lm: Hilfe zur Plot-Funktion für lineare Modelle.  
plot(modell): gibt 6 Diagnose - Grafiken aus.  
# oder plot(modell, which=1), (which=1,2,...,6)  
hist(resid(modell)): Histogramm der Residuen
```

Konfidenzintervall für β_i auswerten:

```
confint(modell, level=0.95)
```

Zur Interpretation der lm-Plots siehe auch:

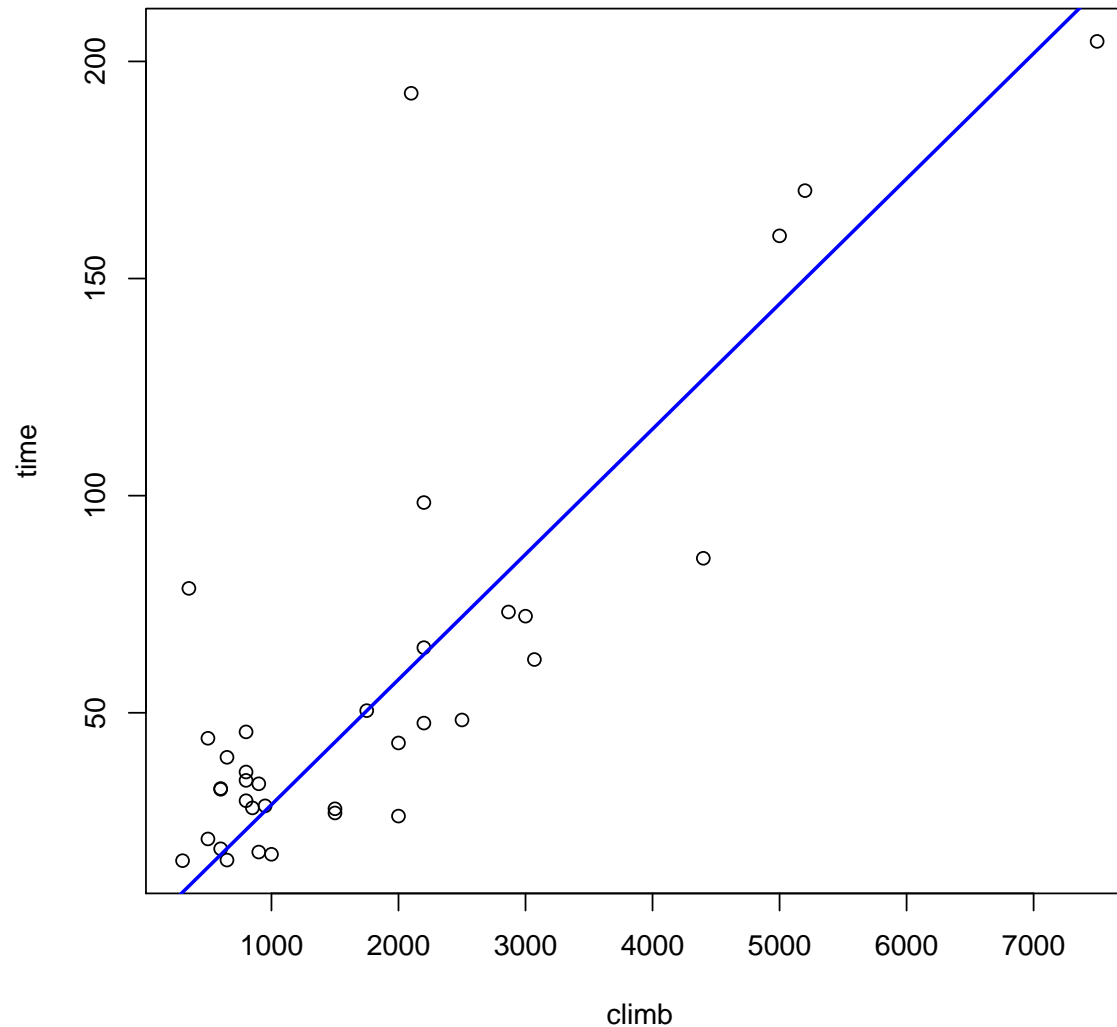
<https://data.library.virginia.edu/diagnostic-plots/>

Beispiel: Modell *time~climb*

```
model2 <- lm(time~climb-1, data=hills)
summary(model2)
anova(model2)
```

Beispiel: Modell $time \sim climb$

```
plot(climb, time, xlab="climb:_feet", ylab="time_minutes")  
abline(model2, lwd = 2, col = "blue")
```



Multiple Linear Regression

Multiples lineares Modell:

$$Y_i = \beta_0 + \beta_1 X_i^1 + \beta_2 X_i^2 + \dots + \varepsilon_i$$

in :

```
model3 <- lm(time~dist+climb-1, data=hills) # hier beta_0 = 0
summary(model3)
anova(model3)
library(car)
Anova(model3)
```


Multiple Lineare Regression: 3D-Plot

Mittels `summary.lm()` bzw. `coef()` erhält man die Koeffizienten für das geschätzte Modell:

$$time = 0.010280(climb) + 5.605651(dist)$$

bzw. bei Angabe in *feet* anstelle von *miles*:

$$1\ mile = 5280\ ft \Rightarrow time = 54.278(climb) + 5.605651(dist)$$

3D-Scatterplots darstellen:

```
install.packages("scatterplot3d")
library(scatterplot3d)
scatterplot3d(hills$dist, hills$climb, hills$time, xlab="dist:_miles",
              ylab="climb:_feet", zlab="time_minutes")
# oder: interaktiver Scatterplot:
install.packages("rgl")
library(rgl)
plot3d(hills$dist, hills$climb, hills$time)
```

Multiple Lineare Regression: 3D-Plot

3D-Scatterplot mit Regressionsebene:

```
g <- scatterplot3d(hills$dist,hills$climb,hills$time)
g$plane3d(model3, lty.box = "solid")

# oder: schöner und mit farbiger Darstellung der z-Achse
grafik3D <- scatterplot3d(hills$dist,hills$climb,hills$time,
                          highlight.3d=TRUE,col.axis=TRUE,
                          col.grid="lightblue",main="3D-Grafik",
                          pch=20,scale.y = 0.5,angle=45,
                          xlab="dist:_miles",ylab="climb:_feet",
                          zlab="time_minutes")

model3 <- lm(time~dist+climb-1,data=hills)
grafik3D$plane3d(model3, lty.box = "solid")
```

Wie gut ist das Modell *model3*?

Die Werte der Test-Statistiken allein können nicht als Maßstab für die Beurteilung der Anpassung dienen.

Beispiel: betrachten wir unsere beste Anpassung für die Daten *hills*:

$$time = 0.010280(climb - feet) + 5.605651(dist - miles)$$

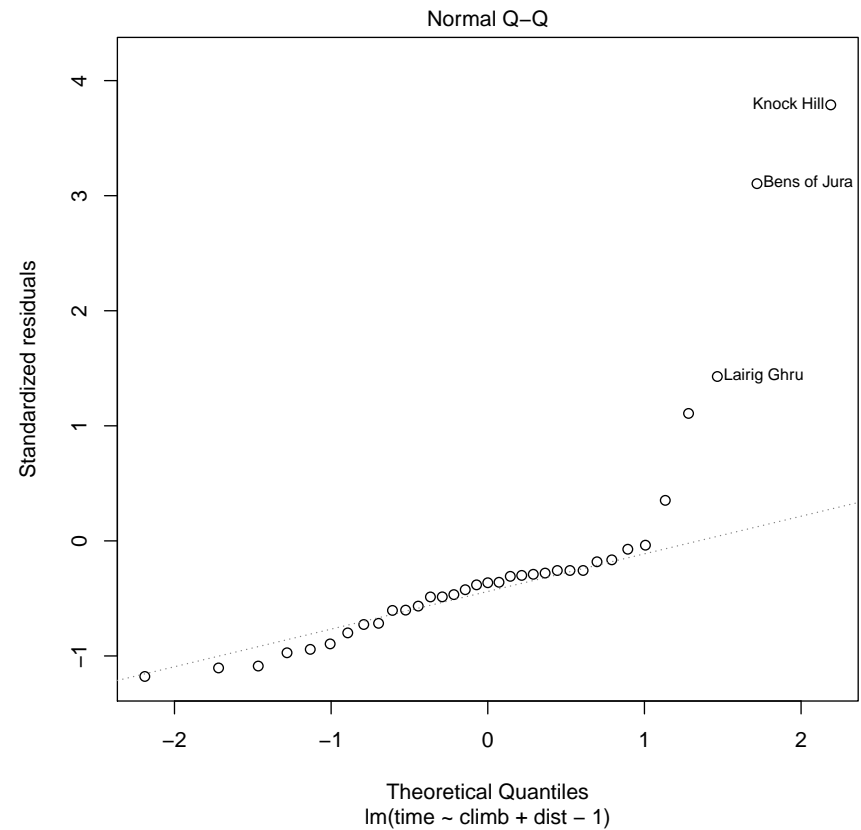
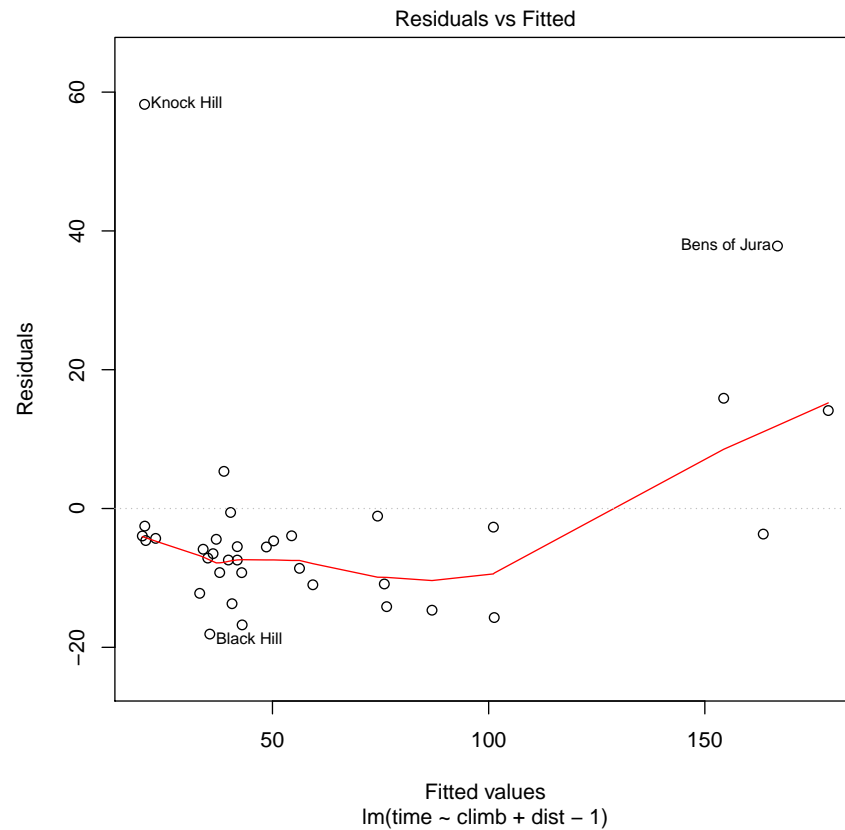
mit $R^2 = 0.96$. (vgl. `summary(model3)`)

Wir betrachten die Verteilung der Residuen mit:

```
plot(model3, which=1)
```

```
plot(model3, which=2)
```

Wie gut ist das Modell *model3*?



Wie gut ist das Modell *model3*?

Die Residuen scheinen nicht normalverteilt zu sein. Möglicherweise gibt es nichtlineare Abhängigkeiten. Man vermutet, dass eventuell die Variable *time* mit den quadratischen Terme von *climb* oder/und *dist* abhängen.

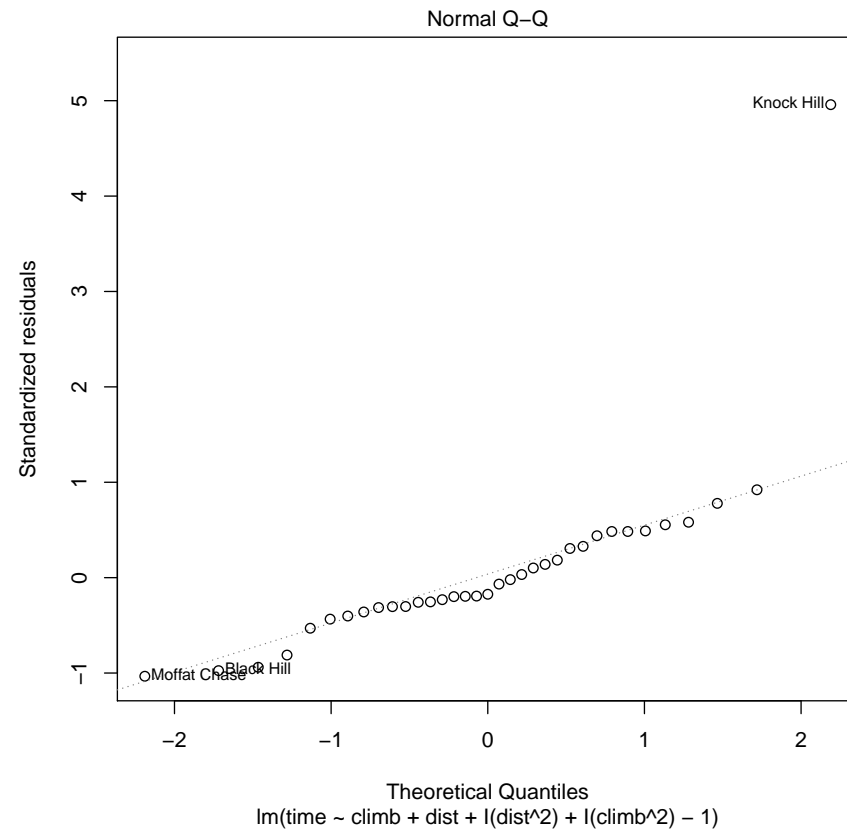
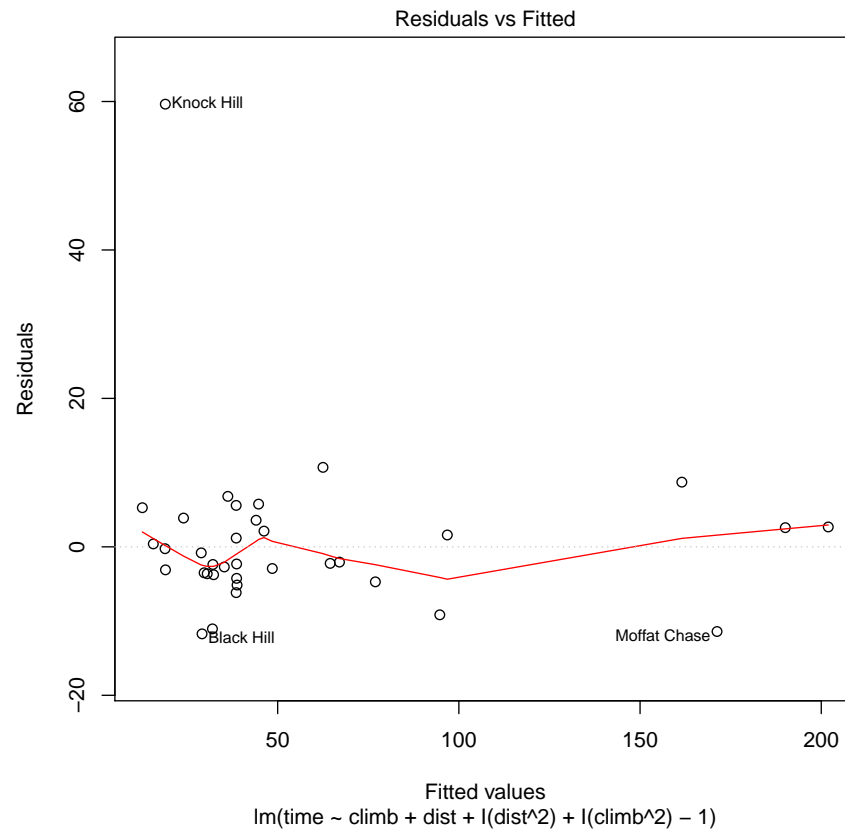
neues Modell:

$$time = \beta_1(climb) + \beta_2(dist) + \beta_3(climb)^2 + \beta_4(dist)^2 + \varepsilon_i$$

```
model4 <- lm(time~dist+climb+I(climb^2)+I(dist^2) -1, data=hills)
summary(model4)
Anova(model4)
```

Welches Modell ist besser?

```
plot(model4, which=1:2)
```



```
anova(model4, model3)
```

Backward Variable Selection

Die *summary* des Modells *model4* und die Diagnoseplots für *model4* deuten auf eine gute Anpassung hin, allerdings sind die Beiträge der Variablen *dist*² und *climb* **nicht signifikant**.

Mittels der Funktion `drop1(model4, test="F")` oder `drop1(model4, test="Chisq")` wird die Wirkung des Entfernens einer Variablen aus *model4* untersucht (*backward variable selection*):

```
drop1(model4, test="F")  
# oder  
drop1(model4, test="Chisq")
```

Backward Variable Selection: Signifikanzkriterium

Single term deletions:

time~dist+climb+l(climb^2)+l(dist^2) -1

	AIC	$Pr(> F)$
<none>	178.75	
dist	200.86	4.53e-06 ***
climb	177.00	0.64098
l(climb^2)	188.57	0.00133 **
l(dist^2)	176.78	0.89546

1. Kriterium: treten unter den $Pr(> F)$ -Werten der reduzierten Modelle Werte auf, die größer als das vorgegebene α sind, so ist der Term mit dem größten P-Wert zu entfernen.

Backward Variable Selection: AIC-Kriterium

Single term deletions:

time~dist+climb+l(climb^2)+l(dist^2) -1

	AIC	$Pr(> F)$
<none>	178.75	
dist	200.86	4.53e-06 ***
climb	177.00	0.64098
l(climb^2)	188.57	0.00133 **
l(dist^2)	176.78	0.89546

2. Kriterium: treten unter den AIC-Werten der reduzierten Modelle Werte auf, die kleiner als der AIC-Wert des vollen Modells sind (none), so ist der Term mit dem kleinsten dieser AIC-Werte zu entfernen.

AIC (Akaike information criterion): mit dem AIC-Wert werden verschiedene Modelle verglichen, die denselben Datensatz beschreiben. Je kleiner der AIC-Wert, desto besser ist ein Modell.

Backward Variable Selection

$I(dist^2)$ entfernen, d.h. das neue Modell ist:

$$time = \beta_1(climb) + \beta_2(dist) + \beta_3(climb)^2 + \varepsilon_i$$

```
model5 <- lm(time~dist+climb+I(climb^2) -1, data=hills)
drop1(model5, test="F")
model6 <- lm(time~dist+I(climb^2) -1, data=hills)
drop1(model6, test="F")
```

Diese Prozedur solange wiederholen, bis keine *AIC*-Werte der reduzierten Modelle kleiner als der *AIC*-Wert des vollen Modells sind (bzw. alle Terme im Modell signifikant sind).

Backward Variable Selection (ebenso *Forward Variable Selection*, bei der das Modell schrittweise um neue Variablen ergänzt wird) läßt sich automatisiert mit der Funktion `step()` ausführen, z.B.:

```
step(model4)
```

Formelbeispiele – lineare Regressionsanalyse

Modell	Formel
$Y = \beta_0 + \beta_1 X + \varepsilon$	$y \sim x$
$Y = \beta_1 X + \varepsilon$	$y \sim x - 1$
$Y = \beta_0 + \varepsilon$	$y \sim 1$
$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$	$y \sim x1 + x2$
$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_{12} X_1 X_2 + \varepsilon$	$y \sim x1 * x2$
$Y = \beta_0 + \beta_{12} X_1 X_2 + \varepsilon$	$y \sim x1 : x2$

Formelbeispiele – nicht-lineare Regressionsanalyse

Modell	Formel
$\log(Y) = \beta_0 + \beta_1 X + \varepsilon$	$\log(y) \sim x$
$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \varepsilon$	$y \sim x_1 + l(x_1^2)$
$Y = \beta_0 + \beta_1 X_1 + \frac{\beta_2}{X_2} + \varepsilon$	$y \sim x_1 + l(1/x_2)$
$\log(Y) = \beta_0 + \beta_1 (X_1 + X_2)^2 + \varepsilon$	$y \sim l((x_1 + x_2)^2)$

Beachte:

$$y \sim x_1 + x_2 : Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$$

$$y \sim l(x_1 + x_2) : Y = \beta_0 + \beta_1 (X_1 + X_2) + \varepsilon$$

Modellselektion & Modellanpassung

Siehe auch:

- `drop1(model, test="F"), add1(model, test="F")`
Bestimme für alle Variablen, die entfernt bzw. hinzugefügt werden können die Auswirkung auf die Anpassungsgüte
- `step(model)`
Automatische Modellselektion (wiederholtes Ausführen von `drop1` bzw. `add1`)
- `update(model)`
Update und Neu-Anpassung eines Modells
- `glm(formula, ...)`
passt ein generalized linear model (GLM) an

Allgemeine Lineare Modelle

für mehrere erklärende Variablen:

$$Y \sim X_1, X_2, \dots, X_m$$

- alle X numerisch: multiple Regression
- alle X kategorial: multifaktorielle ANOVA
- sowohl numerische als auch kategoriale X : ANCOVA (analysis of covariance) bzw. MANCOVA

Lineare und nicht-lineare Modelle

Betrachte das allgemeine Modell:

$$Y \sim f(X_1, X_2, \dots, X_m, \beta_1, \dots, \beta_m)$$

Was heisst hier *linear*?

Linear meint nicht Linearität in den Kovariablen: X_1, \dots, X_m , sondern in den Parametern: β_1, \dots, β_m .

$$Y = \beta_1 X + \beta_2 X^2 + \varepsilon \quad \text{lineares Modell}$$

$$Y = \frac{\beta_1 X}{\beta_2 + X} + \varepsilon \quad (*) \quad \text{nicht-lineares Modell}$$

(*) **Michaelis-Menten-Modell** zur Beschreibung von chemischen Reaktionsraten bei Konzentration X .

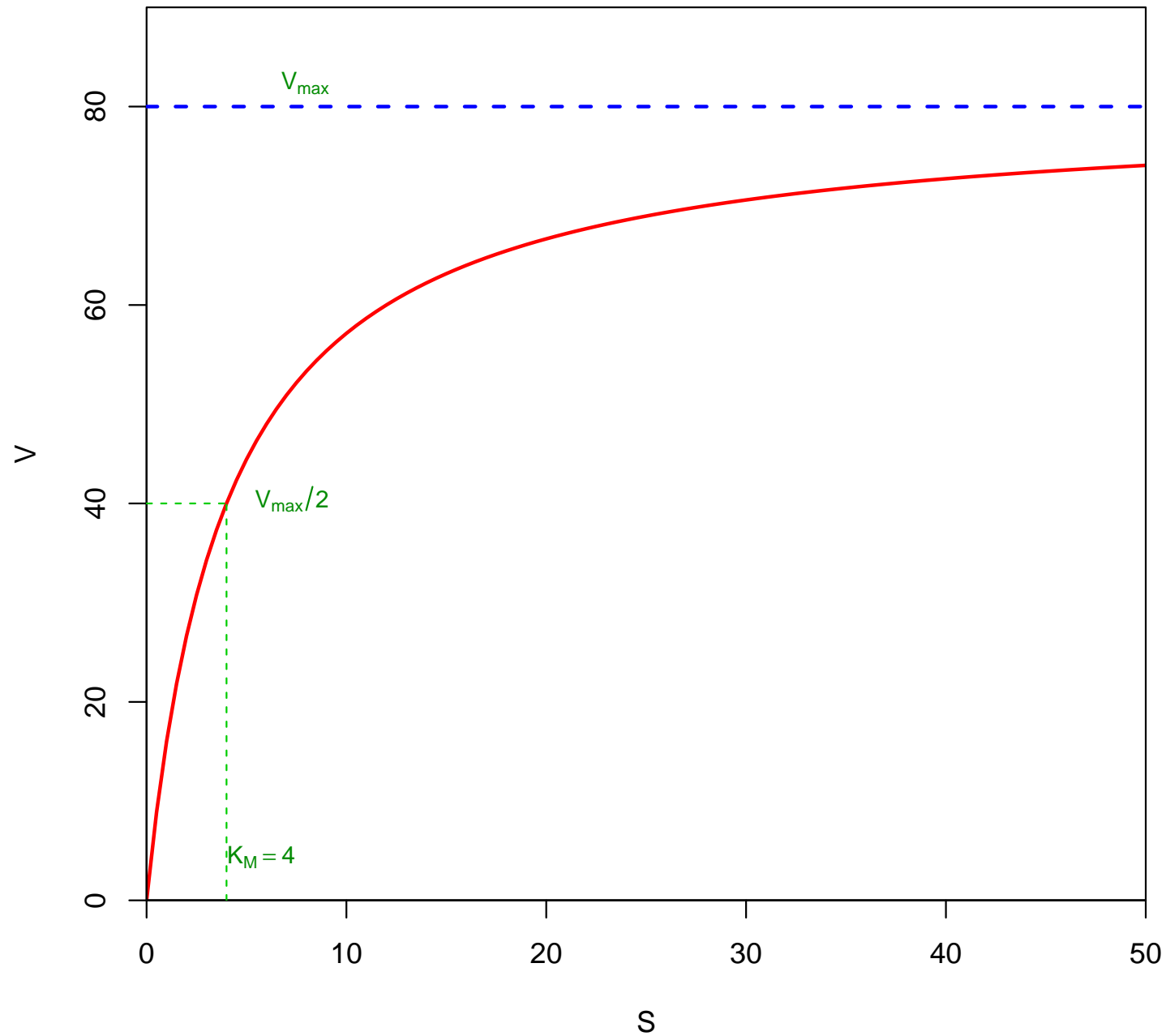
Michaelis-Menten-Modell

Bei vielen Enzymen variiert die Katalysegeschwindigkeit V mit der Substrat-Konzentration S . Im Anfangsbereich nimmt V linear mit S zu. Ab einer gewissen Konzentration beginnt die Kurve abzuflachen und schließlich erreicht sie ihr Maximum v_{max} (Sättigung). In diesem Bereich hat die Konzentration keinen weiteren Einfluß auf V . Das Modell ist definiert durch

$$V = \frac{v_{max}S}{K_M + S}$$

wobei K_M die Michaelis-Konstante ist (setzt man $V = v_{max}/2$, dann hat man $K_M = S$; d.h. K_M stellt die Konzentration dar, bei der das Enzym mit halb-maximaler Geschwindigkeit arbeitet.)

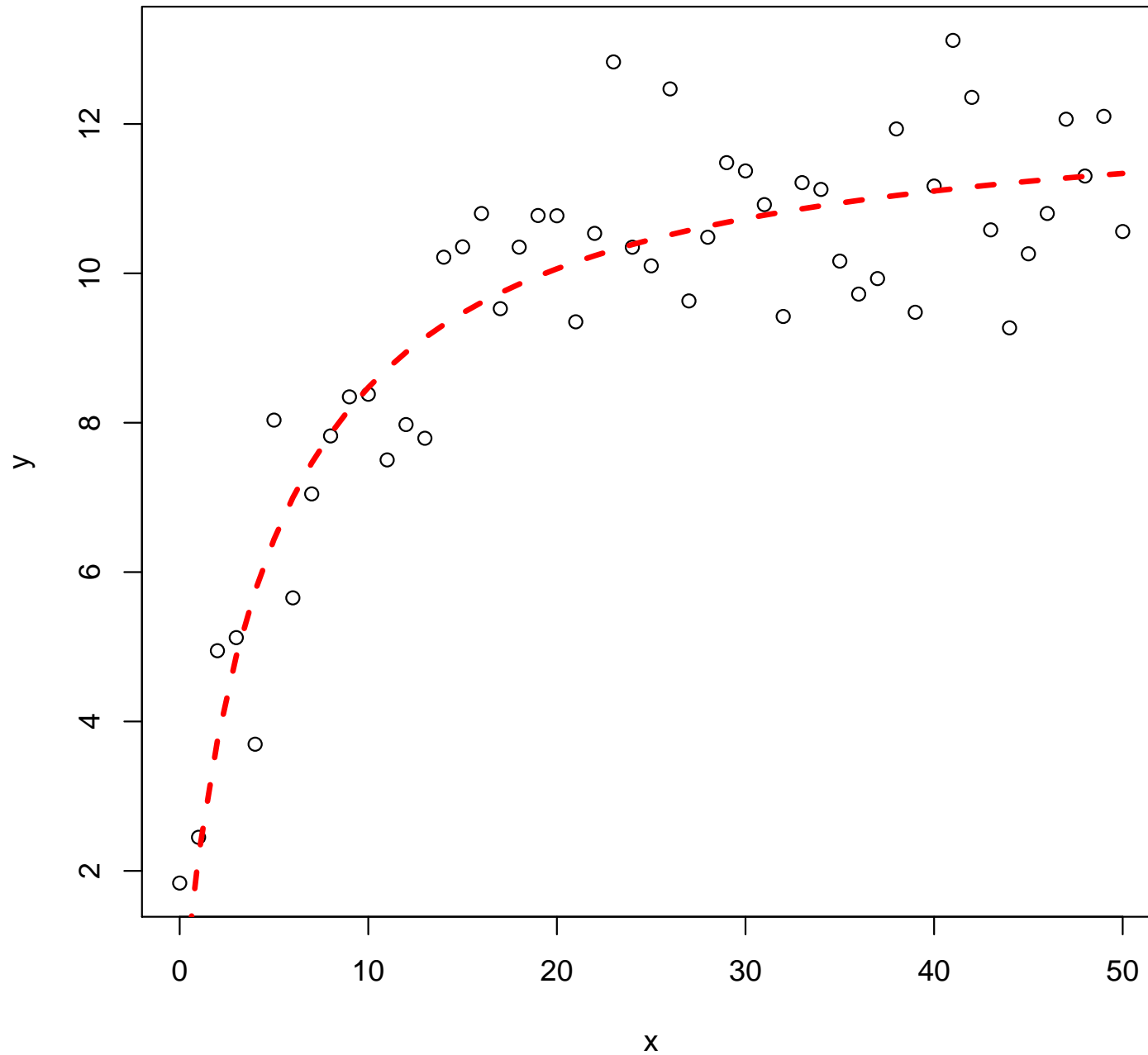
Michaelis-Menten-Modell



Michaelis-Menten-Modell: Beispiel

```
x <- seq(0, 50, 1)
y <- (12*x) / (4+x) + rnorm(51, 0, 1)
# nls Funktion zur Schätzung des Modells
m <- nls(y~a*x/(b+x))
coef(m)
plot(x, y)
lines(x, predict(m), lty=2, col="red", lwd=3)
```

Michaelis-Menten-Modell: Beispiel



Übung: Datenhandling

Gegeben sei folgender artifizieller Datensatz:

```
n <- 200
set.seed(1231)
dat <- data.frame(PatientNR=1:n,
                  BetreuungSehen = rbinom(n,1,0.6),
                  BetreuungHören = rbinom(n,1,0.80),
                  BetreuungParkinson = rbinom(n,1,0.15))
```

Für 200 Senioren (codiert mit einer Patientennummern 1 bis 200) wird angegeben, ob diese gesundheitliche Betreuung zu Sehgesundheit, Hörgesundheit und Parkinson erhalten.

Für eine weitere Befragung sollen zufällig Patienten gezogen werden: Je 20 aus der Patientengruppe mit Betreuung in Sehgesundheit, Hörgesundheit, Parkinson und ebenso je 20 ohne Betreuung – es sind also aus insgesamt 6 Gruppen je 20 Patienten auszuwählen. Ein Patient darf immer nur für eine Gruppe ausgewählt werden.

Übung: Regression

Laden den Datensatz `Boston` aus dem Package `MASS`. Wir betrachten die Variablen `medv` und `lstat`, welche jeweils den Median der Hauswerte und den prozentualen Anteil der Bevölkerung *von niederem Status* in verschiedenen Bostoner Stadtteilen angeben.

Bilden Sie ein geeignetes Regressionsmodell (linear, polynomial, Spline,...) um den Zusammenhang zwischen beiden Variablen zu modellieren und plotten Sie die Regressionskurve.

Zur Lösung vgl.: <http://www.sthda.com/english/articles/40-regression-analysis/162-nonlinear-regression-essentials-in-r-polynomial-and-spline-regression-models/>

Zusatzfolien

- 1 tidy – saubere Daten
- 2 plotly – interaktive Plots
- 3 shiny – Html/JavaScript Frontend für R-Programme

Weiterhin viel Erfolg!

