

Erste Schritte mit R für Praktiker in den Biowissenschaften

V. Kiefel

17. Oktober 2007

Inhaltsverzeichnis

1	Ein-Stichproben-Analyse	4
1.1	Perzentilen, Histogramm	4
1.2	Mittelwert, Median und Standardabweichung	4
2	Berechnung einer Regressionslinie, Korrelationskoeffizient	5
3	Messung der Übereinstimmung der Beurteilung	5
3.1	Cohens Kappa	5
3.2	Spearman's Rangkorrelationskoeffizient	6
4	Kontingenztafeln	6
4.1	Chiquadrat-Test	6
4.2	Prüfung von $k \times 2$ -Feldertafeln auf Trend	7
4.3	“Fisher's exakter Test”	8
4.4	Odds ratio	8
5	Daten von einer Datei einlesen	9
6	Vertrauensgrenzen relativer Häufigkeiten bei binominalverteilter Grundgesamtheit	9
7	Vergleich mehrerer Gruppen	10
7.1	Einfache Varianzanalyse	10
7.2	Bartlett Test	11
8	“Goodness of fit”-Tests (Anpassungstests an bekannte Verteilungen)	12
8.1	Kolmogorov-Smirnov Test für die Güte der Anpassung	12
8.2	Shapiro-Wilk-Test	13
8.3	Überprüfung von Genotyp-Häufigkeiten auf Vorliegen eines Hardy-Weinberg-Gleichweichts	13
9	Vergleich von zwei unabhängigen Stichproben mit dem t-Test	14
10	Vergleich zweier abhängiger Stichproben im t-Test	15
11	Wilcoxon-Test für unabhängige Stichproben	15
12	Friedman-Test	15
13	H-Test (Kruskal-Wallis)	16
14	Überlebenszeit-Analyse	17
14.1	Beispieldaten aus der R-Implementation benutzen	17
14.2	Kaplan Meier-Kurve, Daten aus eigener Datentabelle einlesen	18
15	Statistische Funktionen	19
15.1	Chiquadrat-Verteilung	19
15.2	F-Verteilung	19

15.3	t-Verteilung (Student)	19
15.4	Standardnormalverteilung	20
16	Bestimmung einer notwendigen Stichprobengröße	20
16.1	Vergleich zweier Mittelwerte	20
16.2	Vergleich zweier relativer Häufigkeiten	20
17	Metaanalyse	20
18	Erstellung von Grafiken	21
18.1	Säulendiagramme	22
18.2	Farben und andere Grafikparameter	22
18.3	Boxplots	22
18.4	Scatterplots	23
18.5	Stripchart (eindimensionaler Scatterplot)	24
18.6	Linien verbinden Werte von beliebig vielen Individuen, die zwei oder mehr Zeitpunkten oder Wiederholungen entsprechen	24
18.6.1	Allgemeine Konstruktion mit <code>plot()</code>	24
18.6.2	Konstruktion ähnlicher Diagramme mit <code>interaction.plot()</code>	25
19	Verschiedenes	25
19.1	Berechnung der Determinante einer Matrix	25
19.2	Datumsberechnungen	26
20	Anhang	26
20.1	Hinweise für R unter Linux	26
20.1.1	Installation	26
20.1.2	R kompilieren	27
20.1.3	R konfigurieren	28
20.1.4	PDF-, PostScript- und EPS-Dateien von Abbildungen erstellen.	28
20.2	Allgemeine Befehle	28
20.3	Einstellen von Optionen des Systems	29
20.4	Textdarstellung von R-Objekten als Textdatei speichern und wieder einlesen	29
20.5	Datendateien für R formatieren	29
20.6	Quellcodes	30
20.6.1	ToR.awk	30
20.6.2	Perc.R	30

Vorwort

Das vorliegende Manuskript ist eine kleine willkürlich zusammengestellte Sammlung von “Kochrezepten” für die Anwendung von R^1 mit Verfahren, wie sie von Biowissenschaftlern und Medizinern ständig benötigt werden. Dieser Text wurde in der Absicht geschrieben, dem Benutzer die allerersten Schritte mit R anhand von ihm vertrauten Problemen zu erleichtern. Vor allem benötigt man häufig in der Eile ein Beispiel, anhand dessen man rasch die Eingabe der Daten und die Formulierung einer Auswertung ausführen kann. In diesem Sinne dient dieses Manuskript auch dem Autor als Gedächtnisstütze.

Es empfiehlt sich dringend, beim Ausprobieren der Beispiele zu wichtigen Funktionen die R -Dokumentation zu studieren, um zu erkennen, wie die in diesem Text verwendeten Lösungen in der oft sehr knappen Beschreibung “aussehen”. So wird man dann lernen, alternative und möglicherweise bessere Lösungen zu finden.

Das Manuskript ersetzt nicht die ausgezeichnete Dokumentation zur Beschreibung von R und es ersetzt nicht Lehrbücher oder Originalarbeiten zu statischen Methoden. Vor oder gleichzeitig mit diesem Dokument sollte man sich mit elementaren Datenstrukturen und Funktionen vertraut machen, darunter Vektoren, Listen, Arrays, “data frames”, `read.table()`, `c()`, z. B. in dem Dokument “An introduction to R ” des *R development core teams*, das jeder Installation von R beigelegt ist [1]. Installation und Nutzung der Windows-Version von R dürften weitgehend selbsterklärend sein. Hinweise auf Besonderheiten von R unter Linux finden sich in Abschnitt 20.1.

Für die Richtigkeit der in diesem kleinen Manuskript gemachten Angaben wird keine Gewähr übernommen. In fast allen Fällen ist der beschriebene Weg zur Dateneingabe nicht der einzig mögliche. Bei zunehmender Vertrautheit mit R sollte die Originaldokumentation und Fachliteratur zu den zugrundeliegenden statistischen Verfahren zu Rate gezogen werden! Empfehlenswerte Texte für den Einstieg in elementare Verfahren anhand von R sind [1, 2, 3].

Kritzmow bei Rostock, im Februar 2006
V.K.

¹frei verfügbar unter <http://cran.r-project.org/>

1 Ein-Stichproben-Analyse

1.1 Perzentilen, Histogramm

Zur Berechnung von Perzentilen in der von Reed beschriebenen Form [4] verwende man die Funktion `perc` in `Perc.R` (Quellcode in Abschnitt 20.6.1). Dieser Abschnitt demonstriert die Entwicklung und Verwendung eigener Funktionen.

```
# Aus Reed
hapto <-
c(14, 21, 21, 30, 32, 35, 36, 36, 40, 44, 47, 48, 48, 48, 50, 51, 52, 54, 58,
59, 59, 62, 65, 66, 67, 67, 69, 71, 72, 76, 76, 77, 77, 77, 77, 78, 79, 79,
80, 81, 82, 84, 85, 86, 87, 87, 88, 88, 89, 90, 90, 93, 94, 94, 95, 96, 96,
97, 98, 98, 100, 100, 101, 101, 101, 103, 105, 106, 108, 108, 108, 109, 113,
114, 114, 114, 116, 116, 119, 126, 128, 129, 135, 136, 141, 142, 147, 147,
150, 161, 162, 170, 174, 174, 176, 179, 181, 191, 199, 225)
```

Wenn die Zuweisung der Werte für `hapto` in der Datei `hapto.R` gespeichert ist und das Skript mit der Funktion `perc()` (Abschnitt 20.6.2) in der Datei `perc.R`, dann können Daten und Funktion mit der Funktion `source()` geladen werden und die benutzerdefinierte Funktion `perc()` aufgerufen werden, gefragt sei nach der 97.5-Perzentile:

```
> source("hapto.R")
> source("perc.R")
> perc(hapto, 97.5)
[1] 194.8
>
```

Die eingebaute Funktion von R ergibt einen etwas anderen Wert:

```
> quantile(hapto, 0.975)
97.5%
186.25
>
```

wenn mehr als ein Percentil (Quantil) abgefragt werden soll:

```
> quantile(hapto, c(0.025, 0.5, 0.975))
 2.5%    50%   97.5%
25.275  90.000 186.250
>
```

Ein Histogramm kann mit dem Kommando gezeichnet werden:

```
hist(hapto)
```

Mit eigenen Achsenbeschriftungen, Achsengrenzen versehen:

```
> hist(hapto, xlim=c(0,250), ylim=c(0,25),
+ main="Distribution of Haptoglobin Concentration",
+ xlab="Haptoglobin Concentration", ylab="Frequency")
>
```

1.2 Mittelwert, Median und Standardabweichung

Mittelwert, Median und Standardabweichung sollen zu den Werten in `perc` berechnet werden:

```
> source("hapto.R")
> mean(hapto)
[1] 95.25
> sd(hapto)
[1] 42.71786
> median(hapto)
[1] 90
```

2 Berechnung einer Regressionslinie, Korrelationskoeffizient

Funktion `lsfit`. Die Variablen befinden sich in Vektoren, dann

```
> res <- lsfit(x, y)
> ls.print(res)
```

alternativ:

```
> res <- lm(y ~ x)
> summary.lm(a)
```

Ein Beispiel aus [5, S. 318, Tabelle 107]:

```
> y <- c(12, 17, 11, 13, 16, 14, 15)
> x <- c(13, 17, 10, 17, 20, 11, 15)
> summary.lm(res)
```

Call:

```
lm(formula = y ~ x)
```

Residuals:

```
      1      2      3      4      5      6      7
-1.2694  2.0258 -0.9908 -1.9742 -0.2528  1.5830  0.8782
```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   7.7288      2.8621   2.700  0.0428 *
x              0.4262      0.1897   2.246  0.0746 .
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 1.669 on 5 degrees of freedom

Multiple R-Squared: 0.5023, Adjusted R-squared: 0.4028

F-statistic: 5.046 on 1 and 5 degrees of freedom, p-value: 0.07461

```
>
```

Verschiedene Korrelationskoeffizienten werden mit `cor.test` bestimmt.

3 Messung der Übereinstimmung der Beurteilung

3.1 Cohens Kappa

Daten aus [6, Seite 458]:

```
> library(vcd)
> daten <- c(53, 5, 2, 11, 14, 5, 1, 6, 3)
> mdaten <- matrix(daten, nrow=3, byrow=T)
> mdaten
      [,1] [,2] [,3]
[1,]   53   5   2
[2,]   11  14   5
[3,]    1   6   3
> Kappa(mdaten)
              value          ASE          lwr          upr
```

```
Unweighted 0.4285714 0.08728716 0.2574917 0.5996511
Weighted    0.4923077 0.10057994 0.2951746 0.6894407
>
```

Resultat in [6]: 0,429

3.2 Spearmans Rangkorrelationskoeffizient

Beispiel aus [5, Seite 310]:

```
> a <- c(7, 6, 3, 8, 2, 10, 4, 1, 5, 9)
> b <- c(8, 4, 5, 9, 1, 7, 3, 2, 6, 10)

> cor.test(a, b, method="spearman")

Spearman's rank correlation rho

data: a and b
S = 24, p-value = 0.003077
alternative hypothesis: true rho is not equal to 0
sample estimates:
rho
0.8545455
```

4 Kontingenztafeln

4.1 Chiquadrat-Test

Einzugeben sei die Tafel:

14	22	32
18	16	8
8	2	0

Die Werte in einen Vektor und anschließend in eine Matrix überführen:

```
> r <- c(14, 18, 8, 22, 16, 2, 32, 8, 0)
> rm <- matrix(r, nrow=3)

> rm
      [,1] [,2] [,3]
[1,]  14  22  32
[2,]  18  16   8
[3,]   8   2   0
```

Berechnung mit der Funktion `chisq.test`

```
> chisq.test(rm)

Pearson's Chi-squared test

data: rm
X-squared = 21.5765, df = 4, p-value = 0.0002433

Warning message:
Chi-squared approximation may be incorrect in: chisq.test(rm)
>
```

Eingeben sei die Vierfeldertafel [5, S. 270]:

```
14 85
4 77

> r <- c(15,85,4,77)
> mr <- matrix(r,byrow=T,nrow=2)
> mr
      [,1] [,2]
[1,]  15   85
[2,]   4   77
> chisq.test(mr)

Pearson's Chi-squared test with Yates' continuity correction

data:  mr
X-squared = 3.8107, df = 1, p-value = 0.05093

> chisq.test(mr,correct=F)

Pearson's Chi-squared test

data:  mr
X-squared = 4.8221, df = 1, p-value = 0.02810

>
```

Bei Vierfeldertafeln ist offenbar als Vorgabe Yates' Kontinuitätskorrektur eingestellt, was im zweiten Aufruf abgewählt wurde.

4.2 Prüfung von $k \times 2$ -Feldertafeln auf Trend

Die folgende Tafel soll auf Trend untersucht werden [5, Seite 365]:

	geheilt	
Therapie	nach	
Erfolg	spez.	Anzahl
Score	Therapie	therapiert
1	2	10
2	16	34
3	22	36

Eingabe [2]:

```
> geheilt <- c(2,16,22)
> summe <- c(10,34,36)
> prop.trend.test(geheilt,summe,score=c(1,2,3))

Chi-squared Test for Trend in Proportions

data:  geheilt out of summe ,
      using scores: 1 2 3
X-squared = 5.2197, df = 1, p-value = 0.02233
```

Die gleiche Tafel würde beim Vergleich ohne Berücksichtigungs des Trends keine signifikanten Unterschiede erkennen lassen:

```

> tafel <- c(14, 18, 8, 22, 16, 2)
> tmatrix <- matrix(tafel, byrow = T, nrow = 2)
> tmatrix
      [,1] [,2] [,3]
[1,]  14  18   8
[2,]  22  16   2

> chisq.test(tmatrix)

Pearson's Chi-squared test

data:  tmatrix
X-squared = 5.4954, df = 2, p-value = 0.06407

```

4.3 “Fisher’s exakter Test”

Einzugeben sei die Tafel:

```

10  4
 2  8

```

```

> tf <- c(10, 2, 4, 8)
> mtf <- matrix(tf, nrow=2)
> fisher.test(mtf, alternative="greater")

Fisher's Exact Test for Count Data

```

```

data:  mtf
p-value = 0.01804
alternative hypothesis: true odds ratio is greater than 1
95 percent confidence interval:
 1.435748      Inf
sample estimates:
odds ratio
 8.913675

```

Hier wurde der p -Wert für eine einseitige Fragestellung ermittelt. Für die zweiseitige Fragestellung:

```

> fisher.test(mtf, alternative="two.sided")

Fisher's Exact Test for Count Data

data:  mtf
p-value = 0.03607
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 1.114920 123.433541
sample estimates:
odds ratio
 8.913675

```

4.4 Odds ratio

Eine weitere Möglichkeit zur Berechnung von Konfidenzintervallen einer Odds Ratio besteht in der Verwendung der Funktion `oddsratio` im Paket `epitools` (Beispiel aus [7, Seite 490])

```

> library(epitools)
> tabelle <- matrix(c(24, 96, 48, 592), nrow=2, byrow=T)

```



```

> tabelle
> oddsratio(tabelle,method="midp",conf.level=0.95)
$data
      Outcome
Predictor Disease1 Disease2 Total
Exposed1    24      96    120
Exposed2    48     592    640
Total       72     688    760

$measure
      odds ratio with 95% C.I.
Predictor estimate lower upper
Exposed1  1.000000    NA    NA
Exposed2  3.085417  1.779715  5.236565

$p.value
      two-sided
Predictor midp.exact fisher.exact chi.square
Exposed1    NA          NA          NA
Exposed2  0.0001004225  0.0001119003  1.780411e-05

$correction
[1] FALSE

attr(,"method")
[1] "median-unbiased estimate & mid-p exact CI"
>

```

5 Daten von einer Datei einlesen

Es soll die Ausgleichsgerade zu x - und y - Werten berechnet werden:

	XWERT	YWERT
1	13	12
2	17	17
3	10	11
4	17	13
5	20	16
6	11	14
7	15	15

Die Daten werden dann so eingelesen und ausgewertet:

```

> tafel <- read.table("1.dat", header=TRUE)
> x <- tafel$XWERT
> y <- tafel$YWERT
> res <- lsfit(x, y)
> ls.print(res)

```

6 Vertrauensgrenzen relativer Häufigkeiten bei binominalverteilter Grundgesamtheit

Fragestellung: Man bestimme den 95%-Vertrauensbereich für 7 Treffer unter 20 Beobachtungen.

```

> binom.test(7,20, conf.level=0.95)

```

Exact binomial test

```
data: 7 and 20
number of successes = 7, number of trials = 20, p-value = 0.2632
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.1539092 0.5921885
sample estimates:
probability of success
                0.35

>
```

Eine Alternative zur Eingabe wäre:

```
> binom.test(c(7, 13), conf.level=0.95)
```

Gelegentlich können große Zahlen nicht berechnet werden, dann kann als Alternative `prop.test` verwendet werden.

```
> prop.test(7,20, conf.level=0.95)
```

1-sample proportions test with continuity correction

```
data: 7 out of 20, null probability 0.5
X-squared = 1.25, df = 1, p-value = 0.2636
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.1630867 0.5905104
sample estimates:
      p
0.35
```

Als weitere Alternative steht die Funktion `binom.exact()` aus dem Paket `epitools` zur Verfügung.

7 Vergleich mehrerer Gruppen

7.1 Einfache Varianzanalyse

Folgende Daten sollen in einer einfachen Varianzanalyse untersucht werden. Die Kennungen für die Gruppe müssen Zeichenketten sein. Wenn nicht, muß der Vektor mit den Gruppenkennungen in einen `factor` umgewandelt werden:

```
> gr <- factor(tf$gruppe)
```

Die Eingabe im Einzelnen:

	wert	gruppe
1	6	a
2	7	a
3	6	a
4	5	a
5	5	b
6	6	b
7	4	b
8	5	b
9	7	c

```
10    8    c
11    5    c
12    8    c
```

```
tafel <- read.table("1.dat")
tafel.aov <- aov(tafel$wert~tafel$gruppe)
> summary(tafel.aov)
          Df Sum Sq Mean Sq F value Pr(>F)
tafel$gruppe  2  8.0000  4.0000    3.6  0.071 .
Residuals    9 10.0000  1.1111
---
Signif. codes:  0  '***'  0.001  '**'  0.01  '*'  0.05  '.'  0.1  ' '  1
```

Zwei weitere Möglichkeiten, am Beispiel [5, Seite 388]:

```
Stichprobe 1  2  3
Werte      3  4  8
           7  2  4
           7  6
           3
```

Eingabe der Daten

```
> wert <- c(3,7,4,2,7,3,8,4,6)
> gruppe <- c(1,1,2,2,2,2,3,3,3)
> gruppe<-factor(gruppe)
```

Variante 1:

```
> anova(lm(wert ~ gruppe))
Analysis of Variance Table

Response: wert
          Df Sum Sq Mean Sq F value Pr(>F)
gruppe    2  6.8889  3.4444  0.6889 0.5379
Residuals 6 30.0000  5.0000
```

Variante 2:

```
> oneway.test(wert ~ gruppe,var.equal=T)

One-way analysis of means

data: wert and gruppe
F = 0.6889, num df = 2, denom df = 6, p-value = 0.5379

>
```

7.2 Bartlett Test

Der Bartlett-Test überprüft die Annahme, daß Streuungen in den Gruppen annähernd gleich sind. Die Daten aus [8, Seite 222, 228] werden in die Datei bartlett1.dat eingegeben:

```
Wert    Gruppe
13     1
9       1
```

```

15      1
5       1
25     1
15     1
3       1
9       1
6       1
12     1
42     2
24     2
41     2
19     2
27     2
8       3
24     3
9       3
18     3
9       3
24     3
12     3
4       3
9       4
12     4
7       4
18     4
2       4
18     4

```

Dann erfolgt die Auswertung mit

```

> tafel <- read.table("bartlett1.dat",header=TRUE)
> werte <- tafel$Wert
> gruppen <- factor(tafel$Gruppe)
> bartlett.test(werte, gruppen)

      Bartlett test for homogeneity of variances

data:  werte and gruppen
Bartlett's K-squared = 1.6187, df = 3, p-value = 0.6552

>

```

Die Nullhypothese zur Gleichheit der Varianzen wird demnach nicht abgelehnt.

8 “Goodness of fit”-Tests (Anpassungstests an bekannte Verteilungen)

8.1 Kolmogorov-Smirnov Test für die Güte der Anpassung

Daten aus [9, S. 186]:

```

> da <- c(0.79, 0.68, 0.75, 0.73, 0.69, 0.77, 0.76, 0.74, 0.73, 0.68,
+ 0.72, 0.75, 0.71, 0.76, 0.69, 0.72, 0.70, 0.77, 0.71, 0.74)

> ks.test(da, "pnorm", mean=mean(da), sd=sqrt(var(da)))

      One-sample Kolmogorov-Smirnov test

```

```
data: da
D = 0.0901, p-value = 0.997
alternative hypothesis: two.sided
```

Die Normalverteilungshypothese kann nicht verworfen werden.

8.2 Shapiro-Wilk-Test

Beispiel aus [10, Seite 398-399]: sind die Gewinne an Gewicht von sieben Ratten (70, 85, 94, 101, 107, 118, 132) normalverteilt?

```
> werte <- c(70, 85, 94, 101, 107, 118, 132)
> shapiro.test(werte)
```

```
Shapiro-Wilk normality test
```

```
data: werte
W = 0.998, p-value = 1
```

Es spricht nichts gegen die Annahme, daß die zugrundeliegende Verteilung normalverteilt ist.

8.3 Überprüfung von Genotyp-Häufigkeiten auf Vorliegen eines Hardy-Weinberg-Gleichweichts

Beispiel aus [11]. Beobachtete Aufspaltungsziffern für Br(a/a), Br(a/b), Br(b/b):

Br(a/a)	Br(a/b)	Br(b/b)
1	22	86

```
> library(genetics)
```

```
...
```

```
> hpa5 <- c(rep("A/A", 1), rep("A/B", 22), rep("B/B", 86))
> g1 <- genotype(hpa5)
> summary(g1)
```

```
Number persons typed: 109 (100%)
```

```
Allele Frequency: (2 alleles)
```

```
Count Proportion
A      24      0.11
B     194     0.89
```

```
Genotype Frequency:
```

```
Count Proportion
A/A      1      0.01
B/A     22     0.20
B/B     86     0.79
```

```
Heterozygosity (Hu) = 0.1977574
```

```
Poly. Inf. Content = 0.1767463
```

```
> HWE.chisq(g1, simulate.p.value=FALSE, correct=FALSE)
```

Pearson's Chi-squared test

```
data: tab
X-squared = 0.0985, df = 1, p-value = 0.7536
```

Warning message:

```
Chi-squared approximation may be incorrect in: chisq.test(tab, ...)
>
```

Das gleiche Ergebnis erhält man auch bei manueller Berechnung: Für ein dialleles System bestimmt man die Häufigkeit der Allele (im Br(a/b)-Beispiel ergibt sich $24/(2 * 109)$ für Br(a) und $194/(2 * 109)$ für Br(b)) und berechnet die zu erwartenden Aufspaltungshäufigkeiten, wenn p die Häufigkeit des Allels a und q die Häufigkeit des Allels b ist, gemäß $p^2, 2pq, q^2$, die entsprechenden Häufigkeiten sind 1,32110, 21,35780 und 86,32110. Dann bestimmt man für alle drei Genotypen den Ausdruck $\frac{(B-E)^2}{E}$ (B : beobachtete Häufigkeit, E : erwartete Häufigkeit) und berechnet die Summe dieser drei Werte, die für die Bewertung der Nullhypothese in diesem Beispiel wie χ^2 mit einem Freiheitsgrad verteilt ist [5, Seite 251]. Im Beispiel ergibt das $\chi^2 = 0,09855$. Damit gibt es keinen Anlaß für die Ablehnung der Nullhypothese.

Die Verwendung einer anderen Funktion aus dem `genetics`-Paket ergibt:

```
> HWE.exact(g1)
```

```
Exact Test for Hardy-Weinberg Equilibrium
```

```
data: g1
N11 = 86, N12 = 22, N22 = 1, N1 = 194, N2 = 24, p-value = 1
```

9 Vergleich von zwei unabhängigen Stichproben mit dem t-Test

Die Syntax lautet:

```
> t.test(x, y)
```

Oft befinden sich die Daten beider zu vergleichenden Gruppen in einem Vektor und müssen aus diesem anhand der Gruppenkennungen in einem zweiten Vektor "extrahiert" werden. Beispiel aus P. Armitage und G. Berry: *Statistical Methods in Medical Research*, S 113.

0	1.0
0	1.3
1	1.3
1	1.4
1	1.9
1	2.0
2	2.1
2	2.6
3	2.9

Dateneingabe: der Vektor `werte` enthält die Daten, der Vektor `gr` definiert die Zugehörigkeit zu Gruppen 1, 2:

```
> werte <- c(0,0,1,1,1,1,2,2,3,1,1.3,1.3,1.4,1.9,2,2.1,2.6,2.9)
> gr <- c(rep(1,9), rep(2,9))
> gr <- factor(gr)
> t.test(werte[gr==1], werte[gr==2])
```

```
Welch Two Sample t-test
```

```
data: werte[gr == 1] and werte[gr == 2]
```

```

t = -1.5753, df = 13.844, p-value = 0.1378
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.4440276  0.2218053
sample estimates:
mean of x mean of y
 1.222222  1.833333
>

```

10 Vergleich zweier abhängiger Stichproben im t-Test

Beide Wertereihen werden in Vektoren eingetragen:

```

> a <- c(4, 3.5, 4.1, 5.5, 4.6, 6, 5.1, 4.3)
> b <- c(3, 3, 3.8, 2.1, 4.9, 5.3, 3.1, 2.7)
> t.test(a,b, paired = TRUE)

      Paired t-test

data:  a and b
t = 2.798, df = 7, p-value = 0.0266
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.1781177 2.1218823
sample estimates:
mean of the differences
          1.15

```

11 Wilcoxon-Test für unabhängige Stichproben

Beispiel Sachs 1984 (6. Aufl.) Seite 234:

```

> werte <- c(7,14,22,36,40,48,49,52,3,5,6,10,17,18,20,39)
> gruppe <- c(rep(1,8), rep(2,8))
> gruppe <- factor(gruppe)
> wilcox.test(werte[gruppe==1], werte[gruppe==2], alternative="greater")

      Wilcoxon rank sum test

data:  werte[gruppe == 1] and werte[gruppe == 2]
W = 53, p-value = 0.01406
alternative hypothesis: true mu is greater than 0
>

```

Alternativhypothese: Werte in Gruppe 1 sind größer.

12 Friedman-Test

groups in der Dokumentation sind "Behandlungen", blocks sind Individuen, Stichprobengruppen oder Blöcke. Beide Eingabemöglichkeiten werden demonstriert. Beispiel aus Sachs:

```

Gruppe  1      2      3      4

```

```
Block
  1      27      23      26      21
  2      27      23      25      21
  3      25      21      26      20
```

Die erste Variante ist günstig für die manuelle Eingabe

```
> tafel <- c(27, 27, 25, 23, 23, 21, 26, 25, 26, 21, 21, 20)
> mtafel <- matrix(tafel, nrow=3)
> friedman.test(mtafel)
```

```
Friedman rank sum test
```

```
data: mtafel
Friedman chi-squared = 8.2, df = 3, p-value = 0.04205
```

Diese Form der eingabe ist geeignet für das Auslesen aus einer Datei:

```
> werte <- c(27, 23, 26, 21, 27, 23, 25, 21, 25, 21, 26, 20)
> gruppe <- c(rep(c(1,2,3,4),3))
> gruppe <- factor(gruppe)
> block <- c(rep(1, 4), rep(2, 4), rep(3, 4))
> block <- factor(block)
> friedman.test(werte, gruppe, block)
```

```
Friedman rank sum test
```

```
data: werte, gruppe and block
Friedman chi-squared = 8.2, df = 3, p-value = 0.04205
```

13 H-Test (Kruskal-Wallis)

Beispiel aus Sachs (1984):

A	B	C	D
12.1	18.3	12.7	7.3
14.8	49.6	25.1	1.9
15.3	10.1	47.0	5.8
11.4	35.6	16.3	10.1
10.8	26.2	30.4	9.4
	8.9		

```
> kwdaten <- c(12.1, 14.8, 15.3, 11.4, 10.8)
> kwdaten <- c(kwdaten, 18.3, 49.6, 10.1, 35.6, 26.2, 8.9)
> kwdaten <- c(kwdaten, 12.7, 25.1, 47.0, 16.3, 30.4)
> kwdaten <- c(kwdaten, 7.3, 1.9, 5.8, 10.1, 9.4)
> kwgruppe <- c(rep(1,5), rep(2,6), rep(3,5), rep(4,5))
> kruskal.test(kwdaten, kwgruppe)
```

```
Kruskal-Wallis rank sum test
```

```
data: kwdaten and kwgruppe
Kruskal-Wallis chi-squared = 11.5302, df = 3, p-value = 0.009179
```

alternative Form der Eingabe


```

> gra <- c(12.1, 14.8, 15.3, 11.4, 10.8)
> grb <- c(18.3, 49.6, 10.1, 35.6, 26.2, 8.9)
> grc <- c(12.7, 25.1, 47.0, 16.3, 30.4)
> grd <- c(7.3, 1.9, 5.8, 10.1, 9.4)
> res <- kruskal.test(list(gra, grb, grc, grd))
> res

      Kruskal-Wallis rank sum test

data:  list(gra, grb, grc, grd)
Kruskal-Wallis chi-squared = 11.5302, df = 3, p-value = 0.009179

>

```

14 Überlebenszeit-Analyse

14.1 Beispieldaten aus der R-Implementation benutzen

Der Aufruf der Daten des Datensatzes leukemia aus dem survival-Paket ist in [12, Seite 238–9] beschrieben.

```

> library(survival)
> data(leukemia)

> leukemia
  time status      x
1     9      1 Maintained
2    13      1 Maintained
3    13      0 Maintained
4    18      1 Maintained
5    23      1 Maintained
6    28      0 Maintained
7    31      1 Maintained
8    34      1 Maintained
9    45      0 Maintained
10   48      1 Maintained
11  161      0 Maintained
12    5      1 Nonmaintained
13    5      1 Nonmaintained
14    8      1 Nonmaintained
15    8      1 Nonmaintained
16   12      1 Nonmaintained
17   16      0 Nonmaintained
18   23      1 Nonmaintained
19   27      1 Nonmaintained
20   30      1 Nonmaintained
21   33      1 Nonmaintained
22   43      1 Nonmaintained
23   45      1 Nonmaintained

> leuk.fit <- survfit(Surv(time, status)~x, leukemia)
> summary(leuk.fit)
Call: survfit(formula = Surv(time, status) ~ x, data = leukemia)

              x=Maintained
time n.risk n.event survival std.err lower 95% CI upper 95% CI
  9     11      1   0.909  0.0867   0.7541    1.000
 13     10      1   0.818  0.1163   0.6192    1.000

```

18	8	1	0.716	0.1397	0.4884	1.000
23	7	1	0.614	0.1526	0.3769	0.999
31	5	1	0.491	0.1642	0.2549	0.946
34	4	1	0.368	0.1627	0.1549	0.875
48	2	1	0.184	0.1535	0.0359	0.944

```

x=Nonmaintained
time n.risk n.event survival std.err lower 95% CI upper 95% CI
  5    12     2  0.8333  0.1076   0.6470   1.000
  8    10     2  0.6667  0.1361   0.4468   0.995
 12     8     1  0.5833  0.1423   0.3616   0.941
 23     6     1  0.4861  0.1481   0.2675   0.883
 27     5     1  0.3889  0.1470   0.1854   0.816
 30     4     1  0.2917  0.1387   0.1148   0.741
 33     3     1  0.1944  0.1219   0.0569   0.664
 43     2     1  0.0972  0.0919   0.0153   0.620
 45     1     1  0.0000     NA         NA         NA

```

```

> plot(leuk.fit)
> survdiff(Surv(time, status)~x, data=leukemia, rho=0)
Call:
survdiff(formula = Surv(time, status) ~ x, data = leukemia, rho = 0)

```

	N	Observed	Expected	(O-E)^2/E	(O-E)^2/V
x=Maintained	11	7	10.69	1.27	3.40
x=Nonmaintained	12	11	7.31	1.86	3.40

Chisq= 3.4 on 1 degrees of freedom, p= 0.0653

>

Mit `library(Survival)` wird das Survival-Paket geladen, `data(leukemia)` lädt den Datensatz, `leukemia` zeigt die Daten an. Mit `Surv()` wird ein "Überlebenszeit-Objekt" erstellt, `survfit()` berechnet und `plot()` zeichnet eine Überlebenszeitkurve (n. Kaplan Meier). Mit `survdiff()` wird die Signifikanz der Unterschiede zwischen den Überlebenszeiten² bestimmt. In der Datentabelle ist unter `time` die Überlebenszeit beschrieben, mit `status` wird der *censoring status*³ beschrieben, unter `x` findet sich der Faktor, der die Gruppen definiert ("maintenance chemotherapy given?").

14.2 Kaplan Meier-Kurve, Daten aus eigener Datentabelle einlesen

Nach Daten aus [13, S. 75] soll eine Überlebenszeitkurve erstellt werden. Die geordnete Liste der Beobachtungen (Zeit in Tagen) sei:

1*, 3, 4*, 5, 5, 6*, 7, 7, 7*, 8*

wobei mit * die *censored* Überlebenszeiten ("Ereignis noch nicht eingetreten") seien. Die Daten werden in die Datei `km.dat` eingetragen:

time	status
1	1
2	3
3	4
4	5
5	5
6	6

²bei $\rho=0$ wird der Log-Rank oder Mantel-Haenszel Test, bei $\rho=1$ die Peto & Peto Modifikation des GEHAN-WILCOXON-Tests verwendet

³üblicherweise: 0=noch lebend, 1: verstorben (Ereignis eingetroffen)

```
7 7 1
8 7 1
9 7 0
10 8 0
```

Die Auswertung erfolgt dann mit:

```
> library(survival)
> tafel <- read.table("km.dat", header=T)
> attach(tafel)
> fit <- survfit(Surv(time, status), data=tafel)
> plot(fit)
```

Anmerkung: wenn

```
> attach(tafel)
```

nicht aufgerufen würde, müßte die Lebenszeitanalyse mit

```
> fit <- survfit(Surv(tafel$time, tafel$status), data=tafel)
```

veranlaßt werden.

15 Statistische Funktionen

15.1 Chiquadrat-Verteilung

```
> 1-pchisq(3.84, df=1)
[1] 0.05004352
> 1-pchisq(30.58, df=15)
[1] 0.009993624
```

15.2 F-Verteilung

Berechnung des p -Werts:

```
> 1-pf(9.55, df1=2, df2=3)
[1] 0.05001422
```

Berechnung des F-Werts (gegeben p , $df1$, $df2$):

```
> qf(0.025, df1=16, df2=26, lower.tail=F)
[1] 2.359684
```

15.3 t-Verteilung (Student)

```
> 1-pt(12.706, df=1) # fuer einseitigen Test
[1] 0.0250004
> 2*(1-pt(12.706, df=1)) # fuer zweiseitigen Test
[1] 0.0500008
> 2*(1-pt(2.787, df=25)) # fuer zweiseitigen Test
[1] 0.01001021
```

15.4 Standardnormalverteilung

```
> (1-pnorm(1.64485)) # fuer einseitigen Test
[1] 0.05000037
> 2*(1-pnorm(1.64485)) # fuer zweiseitigen Test
[1] 0.1000007
> 2*(1-pnorm(3.2905, mean=0, sd=1)) # fuer zweiseitigen Test
[1] 0.001000095
```

16 Bestimmung einer notwendigen Stichprobengröße

16.1 Vergleich zweier Mittelwerte

Beispiel 6.3 aus [10]: Vergleich der Lungenfunktion von zwei Gruppen von Männern (mit dem *forced expiratory volume*), zuvor bekannt eine Standardabweichung von 0.5, vorgegeben Signifikanzniveau von 0.05 für den zweiseitigen Test und eine Power von 0.8, wie groß müssen die Stichproben sein für die Erkennung einer Differenz von 0.25:

```
> power.t.test(delta=0.25, sd=0.5, sig.level=0.05, power=0.8, alternative="two.sided" )
```

```
Two-sample t test power calculation

      n = 63.76576
  delta = 0.25
     sd = 0.5
sig.level = 0.05
  power = 0.8
alternative = two.sided
```

NOTE: n is number in *each* group

Damit wird eine Stichprobengröße von 64 pro Gruppe benötigt.

16.2 Vergleich zweier relativer Häufigkeiten

Beispiel 6.5 aus [10]: Vergleich zweier Behandlungen: die eine Behandlung führt zu einer Erfolgsrate von 25%. Wenn eine alternative Behandlungsmethode zu einer Erfolgsrate von 35% führt, wieviel Individuen müssen die zu untersuchenden Gruppen bei einem Signifikanzniveau von 0.05 und einer Power von 90% umfassen?

```
> power.prop.test(p1=0.25, p2=0.35, sig.level=0.05, alternative="two.sided", power=0.9)
```

```
Two-sample comparison of proportions power calculation

      n = 439.2309
     p1 = 0.25
     p2 = 0.35
sig.level = 0.05
  power = 0.9
alternative = two.sided
```

NOTE: n is number in *each* group

also werden pro Gruppe 440 Individuen benötigt.

17 Metaanalyse

Im folgenden wird ein Beispiel aus [14] nachvollzogen. In einer Metaanalyse werden Gruppen von Patienten mit perioperativer autologer Blutspende daraufhin untersucht, wie wahrscheinlich die Gabe von Fremdblut bei einer

Operation sein wird. Hierzu wird das relative Risiko bestimmt. Verglichen werden diese Wahrscheinlichkeiten mit Kontrollgruppen ohne präoperative autologe Blutspende⁴. Die Eingabe in die Datei `blood-tr1.txt`:

```
name      n.trt   n.ctrl  tr.trt  tr.ctrl
busch     239     236    66     133
elawad    45      15     3      14
hedstrom  38      40     7      29
heiss1993 58      62     20     37
heiss1997 29      27     11     13
hoynck    131     137    30     84
kajikawa  10      21     1      13
lorentz   16      15     2      10
```

Die Eingaben⁵:

```
> library(rmeta)
> dl <- read.table("blood-tr1.txt",header=TRUE)
> dl
      name n.trt n.ctrl tr.trt tr.ctrl
1  busch  239   236    66    133
2  elawad  45    15     3     14
3  hedstrom 38    40     7     29
4  heiss1993 58    62    20     37
5  heiss1997 29    27    11     13
6  hoynck  131   137    30     84
7  kajikawa 10    21     1     13
8  lorentz 16    15     2     10
> b <- meta.DSL(n.trt, n.ctrl, tr.trt, tr.ctrl, data=dl,statistic="RR",names=name)
> summary(b)
Random effects ( DerSimonian-Laird ) meta-analysis
Call: meta.DSL(ntrt = n.trt, nctrl = n.ctrl, ptrt = tr.trt, pctrl = tr.ctrl,
  names = name, data = dl, statistic = "RR")
-----
              RR (lower 95% upper)
busch         0.49   0.39   0.62
elawad        0.07   0.02   0.21
hedstrom      0.25   0.13   0.51
heiss1993     0.58   0.38   0.87
heiss1997     0.79   0.43   1.45
hoynck        0.37   0.27   0.53
kajikawa      0.16   0.02   1.07
lorentz       0.19   0.05   0.72
-----
SummaryRR= 0.38 95% CI ( 0.26,0.54 )
Test for heterogeneity: X^2( 7 ) = 22.39 ( p-value 0.0022 )
Estimated random effects variance: 0.14
> plot(b)
>
```

Resultat: in der Gruppe mit präoperativer autologer Spende ist die Wahrscheinlichkeit 0.38 für eine Fremdbluttransfusion verglichen mit der Kontrollgruppe, anders formuliert: die präoperative autologe Spende reduziert das Risiko einer Fremdbluttransfusion um 62%.

18 Erstellung von Grafiken

Im folgenden Abschnitt sind absichtlich die erzeugten Grafiken nicht wiedergegeben, die entsprechenden Quellcodefragmente sollen interaktiv am Prompt eingegeben werden und man sieht dann, was passiert. Gleichzeitig sollte man die Dokumentation öffnen z. B. für Barplot mit `help(barplot.htmlhelp=T)`. Allmählich wird man dann weitergehend probieren können.

⁴n.trt: "n treated", Anzahl der mit autologer Spende "behandelten" Patienten, n.ctrl, Anzahl der Patienten in der Kontrollgruppe, tr.trt: Anzahl der Patienten mit Fremdbluttransfusionen in der Gruppe mit autologer präoperativer Blutspende, tr.ctrl: Anzahl der Patienten mit Fremdbluttransfusionen in der Kontrollgruppe

⁵meta.DSL für *Random effects (DerSimonian-Laird) meta-analysis*

18.1 Säulendiagramme

Es soll ein Säulendiagramm mit den folgenden Werten erstellt werden:

```
> werte <- c(0.03, 0.04, 0.69, 1.25, 0.08, 0.058, 1.1)
```

Es erscheinen senkrechte Säulen in Regenbogenfarben, Farbe wird kontrolliert durch einen Vektor, der col zugewiesen wird

```
> farben <- rep(gray(0.7), 7)
> barplot(werte, col=farben)
```

nun sind alle Säulen hellgrau. Das Diagramm soll waagrechte Balken aufweisen:

```
> barplot(werte, col=farben, horiz=T)
```

Die Säulen sollen schmaler werden.

```
> barplot(werte, col=farben, horiz=T, space=3)
```

Darstellung gruppierter Säulen:

```
> werte <- c(0.03, 0.04, 0.69, 0.09, 1.25, 0.08, 0.58, 0.3)
> wertematrix <- matrix(werte, nrow=2)
> wertematrix
      [,1] [,2] [,3] [,4]
[1,] 0.03 0.69 1.25 0.58
[2,] 0.04 0.09 0.08 0.30
```

```
> barplot(wertematrix, col=farben, horiz=F)
```

Das ergibt "gestapelte Säulen", nebeneinander gruppierte Säulen:

```
> barplot(wertematrix, col=farben, horiz=F, beside=T)
```

als horizontales gruppiertes Diagramm:

```
> barplot(wertematrix, col=farben, horiz=T, beside=T)
```

18.2 Farben und andere Grafikparameter

Dokumentation zu Grafikparametern erhält man mit

```
> help(par)
```

Eine Liste von Farben für col wird mit

```
> colors()
```

gezeigt.

18.3 Boxplots

Annähernd normalverteilte Zufallswerte für 2 Gruppen werden erzeugt mit:

```
> wertel <- rnorm(20, mean=10, sd=2)
> wertel2 <- rnorm(50, mean=16, sd=2.6)
> boxplot(wertel, wertel2)
```

Farbe, Bereich der y-Achse adjustieren

```
> boxplot(wertel, werte2, col="wheat1", ylim=c(0,25))
```

Gruppenbezeichnungen einfügen:

```
> boxplot(wertel, werte2, col="wheat1", ylim=c(0,30),  
+ names=c("group A", "group B"))
```

Körper der Boxen verschieden färben:

```
boxplot(wertel, werte2, col=c("wheat1", "tomato2"),  
+ ylim=c(0,30), names=c("group A", "group B"), horizontal=T)
```

Ein **Boxplot mit logarithmisch unterteilter y-Achse** wird mit dem Zusatz `log="y"` gezeichnet:

```
e <- 2.7182818  
wertel <- rnorm(50, mean=6, sd=0.9)  
wertel <- e**wertel  
werte2 <- rnorm(50, mean=7, sd=1.0)  
werte2 <- e**werte2  
boxplot(wertel, werte2, col="tomato1", log="y", names=c("GR 1", "GR 2"))
```

Oft wird es notwendig sein, die **Boxen in einem Boxplot** zu gruppieren. Dazu kann man die Option `at` nehmen. Im folgenden ein weiteres komplettes Beispiel:

```
werteA1 <- rnorm(20, mean=12, sd=3)  
werteA2 <- rnorm(20, mean=14, sd=3.4)  
werteB1 <- rnorm(18, mean=15, sd=3.2)  
werteB2 <- rnorm(22, mean=18, sd=4)  
werteC1 <- rnorm(24, mean=10, sd=2.2)  
werteC2 <- rnorm(17, mean=9, sd=2)  
boxplot(  
  wertelA1,  
  wertelA2,  
  wertelB1,  
  wertelB2,  
  wertelC1,  
  wertelC2,  
  ylim=c(0,25),  
  at = c(1.2,1.8,3.2,3.8,5.2,5.8),  
  boxwex=0.4,  
  ylab="microgram Ig/ml",  
  names=c("A1", "A2", "B1", "B2",  
          "C1", "C2"), las=2)  
  
abline(v=2.5)  
abline(v=4.5)
```

Um die Positionen der Säulen gleichmäßig zu verteilen, müßte man `at = c(1,2,3,4,5,6)` eintragen. Mit `boxwex` skaliert man die Breite der Boxen. Mit zusätzlichen Linien (`abline(...)`) kann die Gruppierung deutlicher gemacht werden.

18.4 Scatterplots

Im folgenden Beispiel wird ein Plot nach und nach um weitere Punkte, Legenden ergänzt.

```
a <- c(1,2,2.5)  
b <- c(2.6,4,6)
```

```

plot(a,b, xlim=c(0,3), ylim=c(0,8),
     ylab="Y-Werte", xlab="X-Werte", pch=2, col="red")

c <- c(1.1,1.6,1.8,2)
d <- c(2.1,3,3.4,6)
points(c,d, pch=3, type="b", col="blue")

e <- c(1.3,1.7,1.9)
f <- c(2.1,2.7,3.4)

points(e,f, pch=4, type="b", col="magenta")
legend(2.5,1.2,plot=T,legend=c("Group a", "Group c", "Group e"),pch=c(2,3,4),
      col=c("red", "blue", "magenta"))

```

18.5 Stripchart (eindimensionaler Scatterplot)

Während ein Boxplot ein abstrahiertes Bild der Verteilungsform liefert, ist es manchmal auch erwünscht, Originalwerte verschiedener Gruppen direkt zu plotten. Dazu kann man einen Stripchart verwenden.

```

A <- c(1, 3.4, 4, 4.2, 6, 12, 17)
B <- c(4, 8, 12, 13, 14.6, 18.9, 21.4, 14)
stripchart(list(A, B))

```

Gruppennamen werden vergeben und mit anderen Werten für ylim werden die beiden Punktesäulen etwas in die Mitte geholt.

```

stripchart(list(A, B), group.names=c("group A", "group B"), ylim=c(0.5,2.5))

```

Um die Beschriftung der y-Achse horizontal zu setzen und wird die Anweisung `par(las=1)` eingefügt und um den linken Rand zu verbreitern, `par(mar=c(5,6,5,2))`. Damit sieht der Quellcode für die Erstellung des Diagramms so aus:

```

A <- c(1, 3.4, 4, 4.2, 6, 12, 17)
B <- c(4, 8, 12, 13, 14.6, 18.9, 21.4, 14)
stripchart(list(A, B))
par(las=1, mar=c(5,6,5,2))
stripchart(list(A, B), group.names=c("group A", "group B"), ylim=c(0.5,2.5))

```

Weitere Einzelheiten sind in der Dokumentation unter `?stripchart` zu erfragen.

Eine Alternative besteht in der Verwendung von `stripplot` aus dem `lattice`-Paket:

```

library(lattice)
A <- c(1, 3.4, 4, 4.2, 6, 12, 17)
B <- c(4, 8, 12, 13, 14.6, 18.9, 21.4, 14)
trellis.device(color=FALSE)
gruppen <- factor(c(rep("A", length(A)), rep("B", length(B))))
stripplot(c(A,B)~gruppen, ylab="Wert", horizontal=FALSE)

```

18.6 Linien verbinden Werte von beliebig vielen Individuen, die zwei oder mehr Zeitpunkten oder Wiederholungen entsprechen

18.6.1 Allgemeine Konstruktion mit `plot()`

```

par(adj = 0.5,
     xaxt = "n")
plot(
  # erste Linie

```



```

c(1,2), c(2.262,0.014),

type = "b",
xlim = c(0.6,2.4),
ylim = c(0,3),
ylab = "O. D.",
xlab = ""
)

text(1,2.9, vfont=c("sans serif", "plain"), cex=1.4, labels="before absorption")
text(2,2.9, vfont=c("sans serif", "plain"), cex=1.4, labels="after absorption")

# 2. und weitere Linien
lines(type="b", c(1,2), c(0.710,0.214))
lines(type="b", c(1,2), c(0.333,0.041))
lines(type="b", c(1,2), c(1.575,0.042))
lines(type="b", c(1,2), c(2.555,0.063))
lines(type="b", c(1,2), c(2.319,0.283))
lines(type="b", c(1,2), c(0.745,0.063))
lines(type="b", c(1,2), c(0.401,0.023))
lines(type="b", c(1,2), c(0.309,0.052))
lines(type="b", c(1,2), c(0.414,0.040))
lines(type="b", c(1,2), c(0.604,0.042))
lines(type="b", c(1,2), c(0.564,0.021))
lines(type="b", c(1,2), c(0.571,0.043))

lines(type="l", c(0.4,2.6),c(0.2,0.2))

```

Diese Art Darstellung setzt voraus, daß mit `par(xaxt="n")` die Beschriftung der x-Achse ausgeschaltet wird. Dann wird die erste Linie im Plot-Kommando gezeichnet, und dann folgen die weiteren Linien mit `lines()`-Befehlen.

18.6.2 Konstruktion ähnlicher Diagramme mit `interaction.plot()`

Im einfachsten Fall ist die Syntax des Befehls: `interaction.plot(x.factor, trace.factor, response)`. Als Beispiel sei angenommen, daß Werte (values, im Diagramm mit "Intensity" beschriftet) von vier Individuen (subject) zu drei verschiedenen Zeitpunkten (time: t0, t30, t90) untersucht wurden:

```

values = c(10,12,13,8,3,6,15,16,11,1,2,0.3)
time = rep(c("t0", "t30", "t90"), 4)
subject= c(rep("no 1", 3), rep("no 2", 3), rep("no 3", 3), rep("no 4", 3))
interaction.plot(time, subject, values, ylab="Intensity")

```

Ein ähnliches Beispiel ist in [2, Seite 124] besprochen. Dieser Diagrammtyp ist meist geeignet zur Darstellung von Daten, die mit der Friedman-Rangvarianzanalyse untersucht werden.

19 Verschiedenes

19.1 Berechnung der Determinante einer Matrix

Im Beispiel werde für die Berechnung der Determinante

$$D = \begin{vmatrix} 0 & -3 & -8 \\ 1 & 4 & 3 \\ 0 & -15 & -3 \end{vmatrix}$$

die Funktion `det` aus dem Paket `base`.

```
> da <- c(0, -3, -8, 1, 4, 3, 0, -15, -3)
> matda <- matrix(da, nrow=3, byrow=T)
> det(matda)
[1] 111
```

19.2 Datumsberechnungen

Berechnung des Abstands zwischen zwei Daten in Tagen. Beispiel: berechnet werden soll der Abstand in Tagen zwischen 13.12.2000 und 20.4.2001:

```
> library(date)
> mdy.date(4, 20, 2001) - mdy.date(12, 13, 2000)
[1] 128
>
```

Die Differenz beträgt also 128 Tage. Bestimmung des Datums n Tage vor oder nach einem gegebenen Daten: gesucht sei das Datum eines Tages 100 Tage nach dem 16.10.2007:

```
> mdy.date(10, 16, 2007) + 100
[1] 24Jan2008
>
```

Offenbar werden Daten intern als Ganzzahlen mit dem 1.1.1960 als Wert "0" dargestellt:

```
> as.integer(mdy.date(1, 1, 1960))
[1] 0
> as.integer(mdy.date(1, 2, 1960))
[1] 1
>
```

20 Anhang

20.1 Hinweise für R unter Linux

20.1.1 Installation

Das Basisprogramm wird für einige Linux-Distributionen in fertig kompilierter Form angeboten (als rpm-Archiv). Die Installation wird am Shell-Prompt mit einem Kommando wie⁶

```
#rpm -i R-base-x.y.z-m.rpm
```

ausgelöst. Zuvor muß man sich Administratorrechte (*root*) zuweisen mit dem Befehl *su*⁷

Zur Installation der Pakete besorgt man sich von der CRAN-Webseite die entsprechenden Quellcodedateien (.tar.gz) und installiert am Shell-Prompt (z.B. *combinat_0.0-6.tar.gz*) mit *root*-Rechten

```
R CMD INSTALL combinat_0.0-6.tar.gz
```

was funktioniert, wenn sich die notwendigen Entwicklungswerkzeuge einschließlich gcc und Fortran-Compiler auf dem Linux-System befinden. Außerdem sollten sich die Bibliotheken an der Stelle nach einer typischen Installation befinden (Einzelheiten s. Installationsanleitung).

⁶hier beschrieben am Beispiel eines Suse-Linux Systems

⁷Anschließend nicht vergessen, wieder diese Shell mit Administratorrechten mit *exit* zu verlassen!

20.1.2 R kompilieren

Man kann, wenn auf dem Linux-System Entwicklungswerkzeuge (gcc und dazugehörige Entwicklungswerkzeuge einschließlich eines Fortran Compilers, ein T_EX/L_AT_EX-System einschließlich Texinfo und die Entwicklerschnittstelle der readline-Bibliothek) installiert sind, R relativ leicht selbst kompilieren und als lokale “Installation” zum Laufen bringen⁸. Dazu entpackt man (z.B. im eigenen home-Verzeichnis) zunächst (als Nutzer angemeldet) das tar.gz-Archiv mit dem Quellcode⁹

```
gzip -d R-2.4.1.tar.gz
tar -xvf R-2.4.1.tar
```

Dann wechselt man in das entstandene Unterverzeichnis

```
cd R-2.4.1
```

und ruft das configure-Skript auf:

```
./configure
```

Wenn dieses Skript alles gefunden hat, was zum Kompilieren benötigt wird, erscheint eine zusammenfassende Meldung wie:

```
R is now configured for i686-pc-linux-gnu

Source directory:      .
Installation directory: /usr/local

C compiler:           gcc -g -O2
C++ compiler:         g++ -g -O2
Fortran compiler:     gfortran -g -O2

Interfaces supported: X11
External libraries:   readline
Additional capabilities: PNG, iconv, MBCS, NLS
Options enabled:      R profiling

Recommended packages: yes
```

Anschließend ruft man den Befehl zum Kompilieren auf

```
make
```

wobei dieser Vorgang recht lange dauert. Anschließend kopiert man das Shellskript zum Starten von R aus dem bin-Unterverzeichnis in ein Verzeichnis, das in die PATH Umgebungsvariable eingetragen ist. Alternativ zu dieser “lokalen Installation” könnte man R systemweit installieren (als root angemeldet) mit dem Befehl

```
make install
```

Manchen Linux-Distributionen ist kein Fortran-Compiler beigelegt. In diesem Fall kann man auf den freien g95-Compiler zurückgreifen¹⁰.

Beispiel: Ubuntu/Kubuntu Die Vorbereitung zur Einrichtung von R aus den Quellcodes sei näher für die beliebte Ubuntu/Kubuntu-Distribution beschrieben. Mit dem Befehl

```
sudo apt-get install build-essential
```

⁸auf diese Weise benutzt der Autor dieses Manuskripts R auf seinem Linux-System

⁹Der Dateiname des Archivs und das erzeugte Unterverzeichnis haben bei anderen Versionen entsprechend andere Namen

¹⁰<http://g95.sourceforge.net>

wird zunächst die Installation der Standard-Entwicklungswerkzeuge vervollständigt. Anschließend werden mit dem jeweiligen Paketmanager die Pakete `tetex`, `texinfo`, `g77`, `libreadline5-dev`, `xorg-dev` installiert. Anschließend wird wie oben beschrieben das Archiv `R-2.4.1.tar.gz` im Home-Verzeichnis entpackt (womit das Unterverzeichnis `R-2.4.1` entsteht), das `configure`-Skript ausgeführt und das Programm kompiliert. Das Startskript `R` im `bin`-Unterverzeichnis wird in ein Verzeichnis kopiert, dessen Name in der `PATH`-Umgebungsvariablen enthalten ist.

20.1.3 R konfigurieren

Das Verhalten von `R` kann durch den Befehl `options()` beeinflusst werden, dessen Argument eine Liste von Zuweisungen im Format `"variable=wert"` enthält. Will man beispielsweise einen anderen Browser (SeaMonkey) für die Anzeige der Hilfedateien definieren, so kann das (Beispiel für Linux) unter Verwendung des Pfads nach `seamonkey` mit

```
options(browser="/home/username/seamonkey/seamonkey")
```

“von Hand” geschehen, während `R` läuft. Wenn man dagegen eine Konfigurationsdatei anlegen will, um dies vor dem Start des Programms einzutragen, kann das in `Rprofile.site` im `etc`-Unterverzeichnis geschehen. Weitere Optionen zu Konfigurationsdateien sind im Abschnitt “Customizing the environment” im Kapitel “Writing your own functions” des Begleitdokuments “An introduction to `R`” beschrieben.

20.1.4 PDF-, PostScript- und EPS-Dateien von Abbildungen erstellen.

Unter Windows kann eine Grafikdatei mit dem Menü des Grafikfensters gespeichert werden. Um eine **EPS-Datei** mit der Linux-Implementation von `R` zu erstellen, wird die Grafik im Grafikfenster erzeugt mit einem dazu geeigneten Befehl, dann am `R`-Prompt

```
dev.copy2eps()
```

eingegeben. Es wird dann eine Datei mit dem Namen `Rplot.eps` angelegt. Um eine **PostScript-Datei** (Name: `foo.ps`) zu erzeugen, ist vor dem Aufruf des Bildes

```
postscript("foo.ps")
```

einzugeben, das Bild mit dem geeigneten Aufruf zu erzeugen und anschließend mit

```
dev.off()
```

der PostScript-Ausgabeprozess zu schließen. Ähnlich kann eine Grafik in eine PDF-Datei (`aus.pdf`) geschrieben werden:

```
pdf("aus.pdf")
```

Befehle eingeben, um die PDF-Datei zu erzeugen, mit

```
dev.off()
```

den Ausgabeprozess schließen.

20.2 Allgemeine Befehle

Der Befehl `library()` zeigt die Liste der verfügbaren Bibliotheken an, mit `library(MASS)` wird die Library `MASS` geladen. Mit `data()` wird die Liste der Datensätze angezeigt, `data(geyser)` lädt und `geyser` zeigt die Daten zu `geyser`.

Ermitteln des aktuellen Arbeitsverzeichnisses: `getwd()`, wechseln des aktuellen Verzeichnisses: `setwd("LW:/verzeichnis")`. Mit `list.files()` werden die Dateien im aktuellen Verzeichnis angezeigt.

Es empfiehlt sich bei der Arbeit mit `R`, den Quellcode im Texteditor zu bearbeiten und zum Ausführen den abgespeicherten Quellcode auszuführen: `source("scatter1.r")`. Wenn der auszuführende Code ausgedruckt werden soll, geschieht das mit `source("scatter1.r", echo=T)`. Wenn der ausgedruckte Quellcode ohne Prompt erscheinen soll, geben Sie bitte `source("scatter1.r", echo=T, prompt.echo="")` ein.

20.3 Einstellen von Optionen des Systems

Beim Start ist die Anzeige der “nackten” Hilfedateien etwas unkomfortabel. Die “Compiled-HTML-Hilfe” kann als Vorgabe eingestellt werden mit:

```
options(htmlhelp=T)
```

Eine Option kann dann abgefragt werden mit z. B.:

```
getOption('chmhelp')  
[1] TRUE
```

Wenn die Genauigkeit des Systems bei der Ausgabe von Resultaten geändert werden soll, kann das mit

```
options(digits=20)
```

geschehen.

20.4 Textdarstellung von R-Objekten als Textdatei speichern und wieder einlesen

Die Datei `morley.tab` werde eingelesen mit

```
mm <- read.table("morley.tab")
```

mit `mm` können dann die Daten angesehen werden. Der Vektor

```
a <- c(1, 2, 3)
```

und `mm` sollen in eine Textdatellung von R-Objekten geschrieben werden:

```
dump(c("mm", "a"), file = "mm.dat")
```

Mit

```
source("mm.dat")
```

werden dann beide Objekte wieder eingelesen.

20.5 Datendateien für R formatieren

Das folgende `awk`-Skript in Abschnitt 20.6.1 erlaubt die Umwandlung einer Datendatei in das Format für R (es wurde für `gawk`, die GNU-Implementation von `awk` geschrieben).

20.6 Quellcodes

20.6.1 ToR.awk

```
BEGIN {
    i = 1
}

{
    a = $0
    if (i == 1)
    {
        printf("      %s\n", a)
    }
    else if (i > 1)
    {
        if (length(a) > 0)
        {
            printf("%-4i  %s\n", i-1, a)
        }
    }
    i++
}
```

Der Aufruf schreibt den Inhalt von `input.txt` formatiert in `out.dat`.

```
gawk -f ToR.awk input.txt > out.dat
```

benutzt werden.

20.6.2 Perc.R

```
#
# perc(arr, pc.wert) returns the 'pc.wert's percentile of the data
# in the vector 'arr'
# Reference: Reed et al., Clinical Chemistry 1971:17(4);275-84
#
perc <- function(arr, pc.wert) {
    neu <- arr
    neu <- sort(neu)
    n <- length(neu)
    exakt.perz <- (n+1)*pc.wert/100
    hilfswert <- floor(exakt.perz);
    untergr <- neu[hilfswert];
    obergr <- neu[hilfswert + 1];
    if ((hilfswert < 1) || ((hilfswert+1) > n))
    {
        print("Percentile is out of range");
        return(invisible())
    }
    perzentil <- untergr + (obergr - untergr)*(exakt.perz - floor(exakt.perz))
    return(perzentil)
}
```

Literatur

- [1] An Introduction to R. <http://cran.r-project.org/doc/manuals/R-intro.html>.
- [2] Dalgaard P. Introductory Statistics with R. Springer, New York, Berlin, Heidelberg, Hong Kong, London, Milan, Paris, Tokyo, 2002.
- [3] Maindonald J & Braun J, Hg. Data analysis and graphics using R – an example-based approach. Cambridge University press, Cambridge, 2003.
- [4] Reed AH, Berry RJ, & Mason WB. Influence of statistical method on the resulting estimate of normal range. *Clinical Chemistry* 17(4):274–284, 1971.
- [5] Sachs L. Angewandte Statistik. Springer, Berlin, Heidelberg, New York, Tokyo, 6. Aufl., 1984.
- [6] Bortz J, Lienert GA, & Boehnke K, Hg. Verteilungsfreie Methoden in der Biostatistik. Springer-Verlag, Berlin, 1990.
- [7] Sachs L & Hedderich J, Hg. Angewandte Statistik. Methodensammlung mit R. Springer, Berlin, Heidelberg, New York, 12. Aufl., 2006.
- [8] Storm R. Wahrscheinlichkeitsrechnung, mathematische Statistik und statistische Qualitätskontrolle. Fachbuchverlag, Leipzig, Köln, 1995.
- [9] Hartung J, Elpelt B, & Klösener KH. Statistik. R. Oldenbourg, München, Wien, 8. Aufl., 1991.
- [10] Armitage P & Berry G. Statistical methods in medical research. Blackwell Scientific Publications, London, 3. Aufl., 1994.
- [11] Kiefel V, Santoso S, & Mueller-Eckhardt C. The Br(a)/Br(b) alloantigen system on human platelets. *Blood* 73:2219–2223, 1989.
- [12] Krause A & Olson M. Statistics and computing. Springer, New York, Berlin, Heidelberg, 2. Aufl., 2000.
- [13] Matthews DE & Farewell VT. Using and understanding medical statistics. Karger, Basel, 2. Aufl., 1988.
- [14] Henry DA, Carless PA, Moxey AJ, O’Connell D, Forgie MA, Wells PS, & Fergusson D. Pre-operative autologous donation for minimising perioperative allogeneic blood transfusion. *Cochrane database of systematic reviews*: CD003602, 2001.

Index

- Arbeitsverzeichnis
 - Wechsel, 28
- Bartlett Test, 11
- Boxplot, 22
 - logarithmisch unterteilte y-Achse, 23
- Browser für Hilfetexte ändern, 28
- Chiquadrat-Test
 - Kontingenztafeln, 6
- Cohens Kappa, 5
- Datei
 - Daten einlesen, 9
- Daten
 - Differenz zwischen zwei, 26
- Daten aus einer Datei einlesen, 9
- Datumsberechnungen, 26
- Determinate einer Matrix, 25
- Diagramme, 21
- Fishers exakter Test, 8
- Friedman-Test, 15, 25
- Genotyp-Häufigkeiten, 13
- Goodness of fit-Tests, 12
- Grafiken, 21
- H-Test, 16
- Hardy-Weinberg-Gleichweicht, 13
- Histogramm, 4
- `interaction.plot()`, 25
- Kaplan Meier-Kurve, 18
- Kolmogorov-Smirnov Test, 12
- Konfidenzintervall
 - relative Häufigkeiten, 9
- Konfigurieren von R, 28
- Korrelationskoeffizient, 5
- Kruskal-Wallis-Test, 16
- Linux
 - Grafikdateien erstellen, 28
 - Installation unter, 26
- logarithmisch unterteilte y-Achse
 - Boxplot, 23
- Median, 4
- Metaanalyse, 20
- Mittelwert, 4
- Odds Ratio, 8
- Optionen von R einstellen, 29
- Perzentilen, 4
- Quantil, 4
- Rangkorrelationskoeffizient, 6
- `read.table()`, typisches Anwendungsbeispiel (Bartlett Test), 12
- Regressionslinie, 5
- Relative Häufigkeiten
 - Konfidenzintervall, 9
- Säulendiagramm, 22
- Scatterplot, 23
- Shapiro-Wilk-Test, 13
- Spearman's Rangkorrelationskoeffizient, 6
- Standardabweichung, 4
- Stichprobengröße
 - Bestimmung, 20
 - Vergleich zweier Mittelwerte, 20
 - Vergleich zweier relativer Häufigkeiten, 20
- `stripchart()`, 24
- `stripplot()`, 24
- t-Test
 - abhängige Stichproben, 15
 - unabhängige Stichproben, 14
- Übereinstimmung der Beurteilung, 5
- Überlebenszeitdaten, 17
 - Kaplan Meier-Kurve, 18
- Varianzanalyse
 - einfache, 10
- Verteilungen, statistische, 19
 - Chiquadrat-Verteilung, 19
 - F-Verteilung, 19
 - Standardnormalverteilung, 20
 - t-Verteilung, 19
- Verzeichnis
 - Wechsel, 28