Aufbaukurs: Methoden der empirischen Sozialforschung

R-Ergänzungen

Kurt Hornik und Achim Zeileis

2004–02–12

Kapitel 1

Einleitung

1.1 Einfaches Rechnen in R

Zunächst starten wir R: auf den Windows-Rechnern, indem wir unter **Start - Programme - Anwendungen - Mathematik und Statistik** das Programm **Rgui** anwählen (siehe auch „Erste Schritte mit R“ von Christian Neumann), unter Linux indem wir in einem Kommandofenster ‚`R`‘ eingeben.

Mit `setwd()` können wir das Arbeitsverzeichnis geeignet festlegen; mit `getwd()` das verwendete Arbeitsverzeichnis abfragen und dessen Inhalt mit `dir()` anzeigen:

```
R> getwd()

[1] "/home/zeileis"

R> setwd("~/Work/WU/Aufbaukurs/R-Notes/")
R> getwd()

[1] "/home/zeileis/Work/WU/Aufbaukurs/R-Notes"

R> dir()

 [1] "R-Notes.Rnw"          "alcattrneu.tab"
 [3] "alctobac.tab"         "apple.tab"
 [5] "autos.tab"            "bicycle.tab"
 [7] "carfeatures.csv"      "caridex.sav"
 [9] "caridex.tab"          "cereal.csv"
[11] "cereal.sav"           "cmmrcial.csv"
[13] "cmmrcial.sav"         "cmmrcial.tab"
[15] "comphomeneu.tab"      "diet.tab"
```

```
[17] "duxbury.tab"           "election02.csv"
[19] "election02_frequ.csv" "expectedsales.csv"
[21] "lagemasse.tab"         "noewasser.tab"
[23] "promot.csv"            "promot.sav"
[25] "secondhandcar.tab"     "teenagework.tab"
[27] "tv.tab"
```

Wir können R einfach als Rechenmaschine verwenden:

```
R> 1

[1] 1

R> 2

[1] 2

R> 1 + 2

[1] 3

R> 1 - 2

[1] -1

R> 4 * 5

[1] 20

R> 4/5

[1] 0.8

R> 4^2

[1] 16
```

illustriert die Grundrechenarten, und

```
R> sin(0)

[1] 0

R> cos(0)
```

```
[1] 1

R> tan(0)

[1] 0

R> exp(0)

[1] 1

R> log(1)

[1] 0

R> log(0)

[1] -Inf

R> log(exp(1))

[1] 1
```

die trigonometrischen Funktionen, Exponentialfunktion und Logarithmus. Man beachte, dass unendlich kleine beziehungsweise große Werte durch `-Inf` und `Inf` dargestellt werden, und Operationen, die kein wohldefiniertes Ergebnis haben ergeben `NaN` („Not a Number“), e.g.

```
R> 0/0

[1] NaN
```

Man kann Zahlen auch Variablen zuweisen und damit rechnen:

```
R> x <- 1
R> x

[1] 1

R> x + 1

[1] 2

R> y <- x + 1
R> y
```

```
[1] 2
```

```
R> x * y
```

```
[1] 2
```

```
R> x/y
```

```
[1] 0.5
```

Variablen können auch mehr als eine Zahl enthalten: mit Hilfe der Funktion `c()` („combine“) erzeugt man *Vektoren* von Elementen gleichen Typs:

```
R> ALTER <- c(21, 24, 28)
R> ALTER
```

```
[1] 21 24 28
```

```
R> ALTER <- c(ALTER, 21)
R> ALTER
```

```
[1] 21 24 28 21
```

```
R> mean(ALTER)
```

```
[1] 23.5
```

Die Auswahl von Elementen eines Vektors kann durch direkte Angabe der entsprechenden Indizes erfolgen:

```
R> ALTER[2]
```

```
[1] 24
```

```
R> ALTER[1:3]
```

```
[1] 21 24 28
```

```
R> ALTER[-3]
```

```
[1] 21 24 21
```

Dabei bedeutet ein Minus (`-`) „alle Elemente bis auf die mit den angegebenen Indizes“.

Vektoren können nicht nur Zahlen enthalten: mittels

```
R> NAMEN <- c("Hans", "Martha", "Franz-Xaver",
+     "Kunigunde")
R> NAMEN

[1] "Hans"        "Martha"      "Franz-Xaver"
[4] "Kunigunde"
```

erzeugen wir einen Vektor mit Zeichenketten mit den Vornamen der Beobachtungseinheiten deren Alter die entsprechenden Elemente von `ALTER` sind.

Die Funktion `seq` (mit dem Spezialfall `:`) dient zur Erzeugung von Folgen:

```
R> seq(1, 5)

[1] 1 2 3 4 5

R> seq(1, 5, by = 2)

[1] 1 3 5

R> seq(1, 5, length = 9)

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

R> 1:5

[1] 1 2 3 4 5
```

(`a:b` ist eine Abkürzung für `seq(a, b)`).

Mit `rep` kann Werte wiederholen und damit oft die Schreibarbeit deutlich reduzieren:

```
R> rep(1:4, 2)

[1] 1 2 3 4 1 2 3 4

R> rep(1:4, c(2, 2, 2, 2))

[1] 1 1 2 2 3 3 4 4

R> rep(1:4, c(1, 2, 3, 4))

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

Haben wir beispielsweise zunächst 20 Männer und dann 10 Frauen in einer Stichprobe und wollen die Information über das Geschlecht in der Variable `SEX` festhalten, so verwenden wir

```
R> SEX <- rep(c("m", "f"), c(20, 10))
R> length(SEX)

[1] 30

R> SEX

 [1] "m" "m" "m" "m" "m" "m" "m" "m" "m" "m" "m" "m"
[13] "m" "m" "m" "m" "m" "m" "m" "m" "f" "f" "f" "f"
[25] "f" "f" "f" "f" "f" "f"
```

(anstatt den Vektor direkt mit `c` zu erzeugen).

Dabei erkennen wir auch klar die Bedeutung der Zahlen in eckigen Klammern am Zeilenanfang, wenn wir uns den Wert eines Vektors anzeigen lassen: es handelt sich im den Index der unmittelbar folgenden Komponente des Vektors.

1.2 Arbeiten mit Daten in Tabellenform

Sehr oft sind Daten in einem Rechteckschema dargestellt, wobei die Zeilen den Beobachtungen (genauer: Beobachtungseinheiten) und die Spalten den Variablen entsprechen. Oft haben entsprechende Dateien auch noch die Variablennamen in der ersten Zeile. Beispielsweise sind die ersten 5 Zeilen der Datei ‚cmmrcial.tab' (mit den Daten zum Beispiel „Host Selling Commercials — Announcer Commercials") wie folgt:

```
RECALL CEREAL GROUP
     6     FL    SW
     9     CC    SW
     7     KH    SW
     7     CC    SW
```

Dies ist also eine einfache Textdatei, die die durch Tabulatoren separierte Beobachtungen enthält.

Solche Dateien werden vermittels `read.table` eingelesen:

```
R> CMMRCIAL <- read.table("cmmrcial.tab", header = TRUE)
```

Dabei gibt das erste Argument (`"cmmrcial.tab"`) den Namen der Datei an und das zweite, mit einem Namen (`header`) versehene Argument sagt dass in der ersten Zeile der Datei die Variablennamen stehen. Das so erzeugte R Objekt `CMMRCIAL` hat eine spezielle Struktur die das Rechnen mit solchen „rechteckigen“ Datenschemata (sogenannte *data frames*) vereinfacht:

```
R> class(CMMRCIAL)

[1] "data.frame"
```

Vermittels `dim` erhalten wir die „Dimension“ des Datensatzes, also die Anzahl der Beobachtungen und Variablen:

```
R> dim(CMMRCIAL)

[1] 242   3
```

Es handelt sich also im einen Datensatz mit 242 Beobachtungen und 3 Variablen.

Die Namen der Variablen können wir uns mit `names` anzeigen lassen:

```
R> names(CMMRCIAL)

[1] "RECALL" "CEREAL" "GROUP"
```

Um den Vektor von Beobachtungen zu einer Variable, bespielsweise `CEREAL` zu extrahieren, gibt es mehrere Möglichkeiten:

```
R> CMMRCIAL$CEREAL
R> CMMRCIAL[["CEREAL"]]
R> CMMRCIAL[[2]]
R> CMMRCIAL[, "CEREAL"]
R> CMMRCIAL[, 2]
```

(die Ausgabe haben wir der Übersichtlichkeit halber unterdrückt).

Wenn man Teilmengen des Datensatzes bilden will, kann man gleichzeitig Beobachtungen und Variablen selektieren:

```
R> CMMRCIAL[c(1, 8, 22), c("RECALL", "CEREAL")]

   RECALL CEREAL
1       6     FL
8       6     FL
22      6     KH
```

```
R> CMMRCIAL[c(1, 8, 22), 3]

[1] SW SW SW
Levels: NW SW

R> CMMRCIAL[c(1, 8, 22), ]

   RECALL CEREAL GROUP
1       6     FL    SW
8       6     FL    SW
22      6     KH    SW
```

Man sieht dass man Beobachtungen und Variablen jedenfalls durch ihre Positionen ansprechen kann, und die Variablen auch durch ihre Namen. Wenn „nichts“ spezifiziert wird bekommt man „alles“ (Fall 3: alle Variablen).

Oft möchte man aber kompliziertere Teilmengen bilden: angenommen wir brauchen eine Häufigkeitstabelle der Gruppenzugehörigkeiten jener Kinder (Beobachtungseinheiten), die sich für „unser“ Produkt (Canary Crunch, `"CC"`) entschieden haben. Wenn wir dafür

```
R> CMMRCIAL[CEREAL == "CC", "GROUP"]
```

versuchen, erhalten wir die Fehlermeldung ‚`Object "CEREAL" not found`‘. Das liegt daran dass R kein solches Objekt kennt. Wie wir oben gesehen haben, liefert `CMMRCIAL$CEREAL` den Vektor der Beobachtungen von `CEREAL` und daher

```
R> table(CMMRCIAL[CMMRCIAL$CEREAL == "CC", "GROUP"])

NW SW
37 44
```

jedenfalls das gewünschte. Man kann aber auch die Variablen in einem data.frame zu R Objekten machen, indem man den data frame zum sogenannten Suchpfad hinzufügt:

```
R> search()

 [1] ".GlobalEnv"       "package:tools"
 [3] "package:methods"  "package:ctest"
 [5] "package:mva"      "package:modreg"
 [7] "package:nls"      "package:ts"
 [9] "package:fortunes" "Autoloads"
[11] "package:base"
```

```
R> attach(CMMRCIAL)
R> search()

 [1] ".GlobalEnv"       "CMMRCIAL"
 [3] "package:tools"    "package:methods"
 [5] "package:ctest"    "package:mva"
 [7] "package:modreg"   "package:nls"
 [9] "package:ts"       "package:fortunes"
[11] "Autoloads"        "package:base"

R> table(CMMRCIAL[CEREAL == "CC", "GROUP"])

NW SW
37 44
```

Dabei ist das erste Element des Suchpfades (`".GlobalEnv"`) der „Workspace“, in dem standardmäßig Objekte erzeugt werden. Werden diese nicht mehr benötigt kann man sie mit `remove` wieder löschen. Die R Objekte in einem Element des Suchpfades kann man mit `objects` anzeigen:

```
R> objects("CMMRCIAL")

[1] "CEREAL" "GROUP"  "RECALL"
```

Wenn man mit der Analyse eines Datensatzes fertig ist, kann man diesen wieder vom Suchpfad entfernen:

```
R> search()

 [1] ".GlobalEnv"       "CMMRCIAL"
 [3] "package:tools"    "package:methods"
 [5] "package:ctest"    "package:mva"
 [7] "package:modreg"   "package:nls"
 [9] "package:ts"       "package:fortunes"
[11] "Autoloads"        "package:base"

R> detach("CMMRCIAL")
R> search()

 [1] ".GlobalEnv"       "package:tools"
 [3] "package:methods"  "package:ctest"
 [5] "package:mva"      "package:modreg"
 [7] "package:nls"      "package:ts"
 [9] "package:fortunes" "Autoloads"
[11] "package:base"
```

Eine moderne Alternative zu attach/detach bietet die Funktion `subset`:

```
R> table(subset(CMMRCIAL, CEREAL == "CC", GROUP))

NW SW
37 44
```

ist ein weiterer Weg der zum gewünschten Ergebnis führt.

1.3 Import und Export von Daten

Oft liegen Daten nicht als „Rechtecksschemata“ in einfachen Textdateien vor, sondern als Dateien in speziellen binären Formaten, wie zum Beispiel ‚.sav‘ (SPSS) und ‚.xls‘ (Excel) Dateien.

SPSS ‚.sav‘ Dateien kann man mit `read.spss()` aus dem R Erweiterungspaket **foreign** einlesen und via `as.data.frame()` in die übliche Rechtecksform bringen:

```
R> library(foreign)
R> x <- read.spss("cmmrcial.sav")
R> CMMRCIAL <- as.data.frame(x)
R> dim(CMMRCIAL)

[1] 242   4

R> names(CMMRCIAL)

[1] "RECALL"   "CEREAL"   "CMMRCIAL" "CC.YESNO"
```

Excel ‚.xls‘ Dateien kann man nicht direkt einlesen, sondern muss diese von Excel aus in Tabellenform abspeichern. Dazu selektiert man (bei einer deutschsprachigen Version) **Datei - Speichern unter** und wählt dann ‚CSV‘ (Trennzeichen getrennt) im Feld ‚Dateityp‘ aus. Excel speichert dann die aktuelle Tabelle im Trennzeichen-Format mit ‚;‘ als Separator, und man kann diese mit `read.csv2` einlesen. Für englischsprachige Versionen geht man analog vor und verwendet `read.csv`. Es kann immer nur ein Blatt einer Arbeitsmappe im CSV-Format gespeichert werden.

```
R> CMMRCIAL <- read.csv("cmmrcial.csv")
R> dim(CMMRCIAL)

[1] 242   3

R> names(CMMRCIAL)
```

```
[1] "RECALL" "CEREAL" "GROUP"
```

Um Daten abzuspeichern, gibt es prinzipiell auch zwei Möglichkeiten: entweder als Rechtecksschema in einfachen Textdateien oder im eigenen binären Format von R. Ersteres hat den Vorteil, daß diese Art von Dateien von anderen Software-Paketen wie SPSS oder Excel auch eingelesen werden können, letzteres hat den Vorteil, daß R-Objekte (inklusive aller Metainformation wie wir sie in den nächsten Kapiteln noch kennenlernen werden) so wie sie sind abgespeichert und einfach wieder geladen werden können.

Tabellen kann man mit Hilfe der Funktion `write.table` in Textdateien schreiben. Beispielsweise schreibt der Aufruf

```
R> write.table(CMMRCIAL, file = "cmmrcial.csv",
+     sep = ",", row.names = FALSE)
```

den Datensatz `CMMRCIAL` in die Textdatei ‚cmmrcial.csv' mit komma-separierten Werten (ohne Zeilennamen). Um denselben Datensatz in einer binären R-Datei abzuspeichern, wir der Befehl `save` verwendet.

```
R> save(CMMRCIAL, file = "cmmrcial.rda")
```

Der so abgespeicherte Datensatz kann mit Hilfe von `load` wieder geladen werden.

```
R> load("cmmrcial.rda")
```

Dabei ist zu beachten, daß in diesem Fall das Resultat des `load` Aufrufes nicht noch extra einem Objekt zugewiesen werden muß, sondern daß vielmehr all in der entsprechenden Datei gespeicherten Objekte direkt geladen und in der Arbeitsumgebung verfügbar gemacht werden.

Kapitel 2

Ein kategoriales Merkmal

2.1 Wie kann man ein kategoriales Merkmal in R darstellen?

Oft findet man kategoriale Merkmale dargestellt durch numerische Codes, beispielsweise:

```
R> SEX <- c(1, 1, 1, 0, 0, 1, 0, 1, 0, 0)
R> SEX

 [1] 1 1 1 0 0 1 0 1 0 0
```

Dies hat zwei Nachteile: wir brauchen eine Codierungsliste um zu wissen dass etwa ‚1' den Frauen und ‚0' den Männern entspricht, und nichts hindert uns daran die Daten so zu behandeln als ob es sich tatsächlich um qualitative Information handelte:

```
R> sum(SEX)

[1] 5
```

Daher werden kategoriale Merkmale wird in R in speziell durch *Faktoren* dargestellt. Diese kann man explizit durch die Funktion `factor` erzeugen:

```
R> SEX <- factor(SEX, levels = c(0, 1), labels = c("m",
+     "f"))
R> class(SEX)

[1] "factor"

R> SEX
```

```
 [1] f f f m m f m f m m
Levels: m f
```

Nun ist `SEX` ein Objekt der Klasse `"factor"`, die möglichen Kategorien („Levels“) sind `"m"` und `"f"`, und `sum(SEX)` liefert einen Fehler. Die Funktion `summary` liefert eine kurze Zusammenfassung der Daten, im Fall kategorialer Variablen eine Häufigkeitstabelle.

```
R> summary(SEX)

m f
5 5
```

Das obige Beispiel illustriert die Darstellung eines nominalen kategorialen Merkmals: im ordinalen Fall verwendet man entweder `factor` mit dem Argument `ordered = TRUE` oder die Funktion `ordered`:

```
R> GRADES <- ordered(c(1, 5, 3, 3, 3, 4, 4, 5),
+     levels = 1:5, labels = c("A", "B", "C",
+         "D", "F"))
R> class(GRADES)

[1] "ordered" "factor"

R> GRADES

[1] A F C C C D D F
Levels: A < B < C < D < F

R> summary(GRADES)

A B C D F
1 0 3 2 2
```

nimmt die Ergebniscodes einer Prüfung mit einer Notenskala (levels) von 1 bis 5 und vergibt US-amerikanische Beschreibungen (labels). Beachte dass im Fall ordinaler kategorialer Merkmale die Ordnung der Levels angezeigt wird!

Oft muss man sich aber um die explizite Erzeugung kategorialer Merkmale nicht kümmern, da beim Einlesen eines Datensatzes vermittels `read.table` nicht-quantitative Variable standardmäßig auf Faktoren umgewandelt werden:

```
R> CMMRCIAL <- read.table("cmmrcial.tab", header = TRUE)
R> CMMRCIAL$CEREAL[1:10]
```

```
 [1] FL CC KH CC BB BB FL FL BB KH
Levels: BB CC FL KH
```

Allerdings ergibt diese Umwandlung natürlich immer nur nominale Variablen mit alphabetisch geordneten Kategorien. Die Umwandlung in ein ordinales kategoriales Merkmal oder die Neuordnung der Kategorien erfordert zusätzliche Aufmerksamkeit, bspw. durch einen Aufruf von `factor`.

```
R> CMMRCIAL$CEREAL <- factor(CMMRCIAL$CEREAL,
+     levels = c("CC", "KH", "BB", "FL"))
R> CMMRCIAL$CEREAL[1:10]

 [1] FL CC KH CC BB BB FL FL BB KH
Levels: CC KH BB FL
```

Grundsätzlich ist es daher von Vorteil die Daten in einem Format zu speichern, in dem entsprechende Information nicht verloren geht.

Oft entstehen kategoriale Merkmale durch Intervalleinteilung metrischer Variablen, wofür es in R die Funktion `cut` gibt. Beispielsweise entspricht

```
R> GRADES <- cut(c(0, 12, 24), c(-1, 11, 17,
+     21, 23, 24))
R> GRADES

[1] (-1,11] (11,17] (23,24]
Levels: (-1,11] (11,17] (17,21] (21,23] (23,24]
```

der üblichen Notenskala anhand von Testergebnissen mit maximal 24 Punkten. Beachte dass die Intervalle standardmäßig links offen und rechts geschlossen sind (daher auch ‚-1‘). Wer jetzt noch die Kategorien US-amerikanisch umbennen will kann dies wie folgt tun:

```
R> levels(GRADES) <- c("A", "B", "C", "D", "F")
R> GRADES

[1] A B F
Levels: A B C D F
```

„Aufräumen“ durch entfernen der nicht mehr verwendeten Objekte:

```
R> remove(SEX, GRADES, CMMRCIAL)
```

2.2 Wie kann man ein kategoriales Merkmal numerisch beschreiben?

Die absoluten Häufigkeiten berechnet man mit der Funktion `table`; für die relativen Häufigkeiten dividiert man die absoluten durch die Anzahl der Beobachtungen (also die *Länge* des Datenvektors oder die Summer der absoluten Häufigkeiten); wer die relativen Häufigkeiten in Prozent möchte muss noch mit 100 multiplizieren.

Beispiel 2.1 (Präsentatorenwerbung und normale Werbung)

Zunächst den Datensatz einlesen und dem Suchpfad hinzufügen:

```
R> CMMRCIAL <- read.table("cmmrcial.tab", header = TRUE)
R> attach(CMMRCIAL)
```

Nun liefert

```
R> table(CEREAL)
```

```
CEREAL
BB CC FL KH
64 81 45 52
```

die absoluten und

```
R> table(CEREAL)/length(CEREAL)
```

```
CEREAL
       BB        CC        FL        KH
0.2644628 0.3347107 0.1859504 0.2148760
```

die relativen Häufigkeiten.

Wenn man absolute und relative Häufigkeiten als Spalten einer Tabelle nebeneinander haben möchte, verwendet man `cbind` (für *column binding*, es werden also die Spalten zusammengefügt):

```
R> absH <- table(CEREAL)
R> relH <- absH/length(CEREAL)
R> cbind(absH = absH, relH = relH)
```

```
   absH      relH
BB   64 0.2644628
CC   81 0.3347107
FL   45 0.1859504
KH   52 0.2148760
```

Schließlich sind wir mit den Berechnungen für den Datensatz fertig und können „zusammenräumen“:

```
R> detach("CMMRCIAL")
R> remove(CMMRCIAL)
```

Kumulierte Häufigkeiten berechnen wir durch Bildung von Teilsummen vermittels `cumsum`.

Beispiel 2.2 (Canada Social Survey, 1991)

Hier haben wir keine Rohdaten und erzeugen die Tabelle direkt:

```
R> absH <- as.table(c(keine = 2310, gering = 3783,
+     hoch = 4397, "sehr hoch" = 844))
R> absH
```

```
    keine    gering      hoch sehr hoch
     2310      3783      4397       844
```

Für die relativen Häufigkeiten dividieren wir durch die Summe der absoluten Häufigkeiten:

```
R> relH <- absH/sum(absH)
R> relH
```

```
     keine     gering       hoch  sehr hoch
0.20381154 0.33377448 0.38794777 0.07446621
```

Für die kumulierten relativen Häufigkeiten berechnen wir die Teilsummen:

```
R> cumsum(relH)
```

```
[1] 0.2038115 0.5375860 0.9255338 1.0000000
```

Alles übersichtlich in einer Tabelle:

```
R> cbind("abs H" = absH, "rel H" = relH, "kum rel H" = cumsum(relH))
```

```
          abs H      rel H kum rel H
keine      2310 0.20381154 0.2038115
gering     3783 0.33377448 0.5375860
hoch       4397 0.38794777 0.9255338
sehr hoch   844 0.07446621 1.0000000
```

Und schließlich zusammenräumen:

```
R> remove(absH, relH)
```

2.3 Wie kann man ein kategoriales Merkmal grafisch beschreiben?

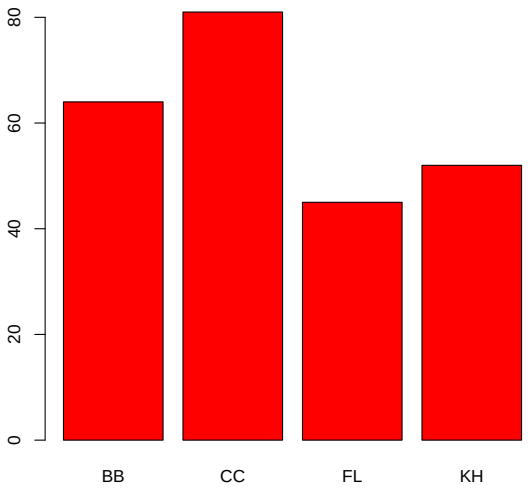
Balken- und Kreisdiagramme zur grafischen Beschreibung eines kategorialen Merkmals werden mit `barplot` beziehungsweise `pie` erzeugt.

Wir illustrieren dies wieder an den präferierten Cerealien: zunächst die Daten einlesen und `CEREAL` tabellieren:

```
R> CMMRCIAL <- read.table("cmmrcial.tab", header = TRUE)
R> attach(CMMRCIAL)
R> tabCer <- table(CEREAL)
```

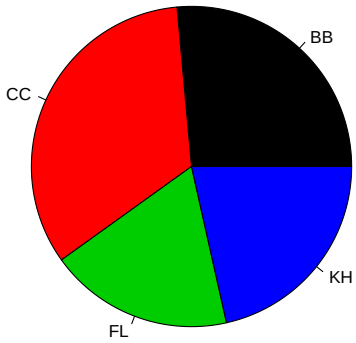
Ein Balkendiagramm mit `barplot`:

```
R> barplot(tabCer)
```



Ein Kreisdiagramm mit `pie`:

```
R> pie(tabCer, col = 1:4)
```



Und zusammenräumen:

```
R> detach("CMMRCIAL")
R> remove(CMMRCIAL)
```

2.4 Kommen alle Kategorien gleich häufig vor?

Mittels der Funktion `chisq.test` kann man verschiedene χ^2-Tests für Häufigkeitstabellen ausführen. Der χ^2-*Test auf Gleichverteilung* ist der einfachste Fall: hier muss als einziges Argument die Tabelle der absoluten Häufigkeiten (aber *nicht* die Rohdaten) spezifiziert werden:

```
R> CMMRCIAL <- read.table("cmmrcial.tab", header = TRUE)
R> attach(CMMRCIAL)
R> chisq.test(table(CEREAL))

        Chi-squared test for given probabilities

data:  table(CEREAL)
X-squared = 12.314, df = 3, p-value = 0.006381
```

Die Ausgabe ist eine kompakte Zusammenfassung der Inferenzergebnisse: welcher Test mit welchen Daten verwendet wurde; die Werte der Teststatistik, die Anzahl der Freiheitsgrade, und der p-Wert.

Wenn wir das Ergebnis des Tests einem R Objekt zuweisen, so können wir erkennen, dass dieses von einem speziellen Typ (eine Liste der Klasse `"htest"`) ist und die Resultate in geeigneten Komponenten stehen:

```
R> testErg <- chisq.test(table(CEREAL))
R> typeof(testErg)

[1] "list"

R> class(testErg)

[1] "htest"

R> names(testErg)

[1] "statistic" "parameter" "p.value"   "method"
[5] "data.name" "observed"  "expected"  "residuals"
```

Insbesondere erhalten wir beobachtete und unter der Nullhypothese der Gleichverteilung erwartete Häufigkeiten durch Extraktion der Komponenten `observed` und `expected`:

```
R> rbind("beobachtete H" = testErg$observed,
+     "erwartete H" = testErg$expected)

                BB   CC   FL   KH
beobachtete H 64.0 81.0 45.0 52.0
erwartete H   60.5 60.5 60.5 60.5
```

Für die Berechnung von kritischen Werten ersetzt die Funktion `qchisq` das Nachschlagen in der Tabelle der χ^2-Verteilung:

```
R> qchisq(0.95, df = 3)

[1] 7.814728
```

ergibt den kritischen Wert von 7.815, der kleiner als der Wert der Teststatistik ist.

In jedem Fall kann die Nullhypothese, dass alle Kategorien gleich häufig vorkommen, verworfen werden.

2.5 Entsprechen die Häufigkeiten bestimmten Vorgaben?

Um zu testen ob Häufigkeiten in der Grundgesamtheit einer bestimmten Vorgabe (nicht notwendigerweise der Gleichverteilung) folgen, wendet man wieder `chisq.test` auf die beobachteten Häufigkeiten an und spezifiziert die vorgegebenen Häufigkeiten mittels des Argumentes `p`.

Beispiel 2.3 (Repräsentativität einer Meinungsumfrage)

Hier haben wir keine Rohdaten und erzeugen die Tabelle direkt:

```
R> absH <- as.table(c(LGo = 23, LGm = 133, Mu = 39,
+     Va = 5))
R> absH

LGo LGm  Mu  Va
 23 133  39   5
```

Der Test ergibt:

```
R> chisq.test(absH, p = c(35.8, 51.7, 10.8, 1.7)/100)

        Chi-squared test for given probabilities

data:  absH
X-squared = 56.2314, df = 3, p-value = 3.749e-12
```

und zeigt, dass die Stichprobe nicht repräsentativ ist.

Interessant ist wieder ein Vergleich von beobachteten und erwarteten Häufigkeiten:

```
R> testErg <- chisq.test(absH, p = c(35.8, 51.7,
+     10.8, 1.7)/100)
R> rbind("beobachtete H" = testErg$observed,
+     "erwartete H" = testErg$expected)

               LGo   LGm   Mu  Va
beobachtete H 23.0 133.0 39.0 5.0
erwartete H   71.6 103.4 21.6 3.4
```

zeigt, dass in der Stichprobe zuwenig Lebensgemeinschaften ohne und zuviele mit Kindern enthalten sind.

2.6 In welchem Bereich kann man einen Anteil erwarten?

Mit der Funktion `binom.test` kann man Inferenz über Anteile treiben: sie liefert sowohl Konfidenzintervalle als auch Tests von Hypothesen.

Beispiel 2.4 (Nationalratswahl 2002)

Wir betrachten nur die 325 Befragten, die die Sonntagsfrage beantwortet haben. Von diesen haben sich 105 für die SPÖ entschieden, das entspricht etwa 32.3%. Damit erhalten wir

```
R> binom.test(105, 325)

        Exact binomial test

data:  105 and 325
number of successes = 105, number of trials =
325, p-value = 1.661e-10
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.2725143 0.3769005
sample estimates:
probability of success
             0.3230769
```

und somit das gesuchte 95%-Konfidenzintervall für den Anteil von $[0.273, 0.377]$ (und `probability of success` bezeichnet nochmals den Stichprobenanteil von 32.3%).

Für ein Konfidenzintervall mit 99%-iger Sicherheit verwenden wir

```
R> binom.test(105, 325, conf.level = 0.99)$conf.int

[1] 0.2577561 0.3937006
attr(,"conf.level")
[1] 0.99
```

um ein Intervall von $[0.258, 0.394]$ zu erhalten. Beide Intervalle enthalten auch das wahre Abschneiden von 36.51%.

Kapitel 3

Mehrere kategoriale Merkmale

3.1 Wie kann man zwei kategoriale Merkmale numerisch beschreiben?

Kontingenztafeln (Kreuztabellen) erzeugt man entweder wiederum mit `table`

```
R> CMMRCIAL <- read.table("cmmrcial.tab", header = TRUE)
R> attach(CMMRCIAL)
R> tab <- table(CEREAL, GROUP)
R> tab

      GROUP
CEREAL NW SW
    BB 31 33
    CC 37 44
    FL 26 19
    KH 27 25
```

oder mit `xtabs`:

```
R> xtabs(~CEREAL + GROUP, data = CMMRCIAL)

      GROUP
CEREAL NW SW
    BB 31 33
    CC 37 44
    FL 26 19
    KH 27 25
```

Das erste Argument von `xtabs` ist eine sogenannte „Formel“ (`formula`), welche man so lesen kann: die Variablen auf der linken Seite werden modelliert/erklärt durch (`~`) die Variablen auf der rechten Seite. Nun gibt es hier keine Variablen auf der linken Seite und daher wird die Kreuztabelle gebaut, die durch die Variablen `CEREAL` und (`+`) `GROUP` definiert wird. Die Variablen werden aus dem Datensatz `CMMRCIAL` (spezifiziert durch das `data` Argument) genommen.

Die Randinformation erzeugt man aus Tabellen durch Bildung von Zeilen- beziehungsweise Spaltensummen. In R am einfachsten mit Hilfe der Funktion `apply` (mit der man beliebigdimensionale Ränder von beliebigdimensionalen Feldern berechnen kann): da in einer Matrix der Zeilenindex der erste und der Spaltenindex der zweite ist, gibt

```
R> apply(tab, 1, sum)

BB CC FL KH
64 81 45 52
```

die Zeilen- und

```
R> apply(tab, 2, sum)

 NW  SW
121 121
```

die Spaltensummen. Für Zeilen- und Spaltensummen von Matrizen (d.h. 2-dimensionalen Feldern) gibt es auch die Bequemlichkeitsfunktionen `rowSums` und `colSums`:

```
R> rowSums(tab)

BB CC FL KH
64 81 45 52

R> colSums(tab)

 NW  SW
121 121
```

Zur Berechnung der bedingten Information muss man die Elemente in den jeweiligen Zeilen (Spalten) durch die entsprechenden Zeilensummen (Spaltensummen) teilen. Auch hier gibt es eine allgemeine Funktion, `sweep`:

```
R> rowProp <- sweep(tab, 1, apply(tab, 1, sum),
+     "/")
R> rowProp
```

```
      GROUP
CEREAL NW        SW
    BB 0.4843750 0.5156250
    CC 0.4567901 0.5432099
    FL 0.5777778 0.4222222
    KH 0.5192308 0.4807692
```

berechnet bedingte relative Häufigkeiten für die Zeilen

```
R> colProp <- sweep(tab, 2, apply(tab, 2, sum),
+     "/")
R> colProp

      GROUP
CEREAL NW        SW
    BB 0.2561983 0.2727273
    CC 0.3057851 0.3636364
    FL 0.2148760 0.1570248
    KH 0.2231405 0.2066116
```

bedingte relative Häufigkeiten für die Spalten (um Zeilenprozent beziehungsweise Spaltenprozent zu erhalten muss man noch mit 100 multiplizieren). Zur Probe:

```
R> apply(rowProp, 1, sum)

BB CC FL KH
 1  1  1  1

R> apply(colProp, 2, sum)

NW SW
 1  1
```

Für schreibfaule Menschen gibt es wieder zwei Bequemlichkeitsfunktionen, die die obigen Operationen in jeweils einen Aufruf zusammenfassen: `margin.table` zur Bildung von Randhäufigkeiten und `prop.table` zur Berechnung von bedingten relativen Häufigkeiten:

```
R> margin.table(tab, 1)

CEREAL
BB CC FL KH
64 81 45 52
```

```
R> prop.table(tab, 1)

      GROUP
CEREAL NW        SW
    BB 0.4843750 0.5156250
    CC 0.4567901 0.5432099
    FL 0.5777778 0.4222222
    KH 0.5192308 0.4807692
```

liefert die Randhäufigkeiten von `CEREAL` und die bedingte Verteilung von `GROUP`.

3.2 Wie kann man zwei kategoriale Merkmale grafisch beschreiben?

Gruppierte Balkendiagramme erzeugt man mit der Funktion `barplot`, indem man das Argument `beside` auf `TRUE` setzt.

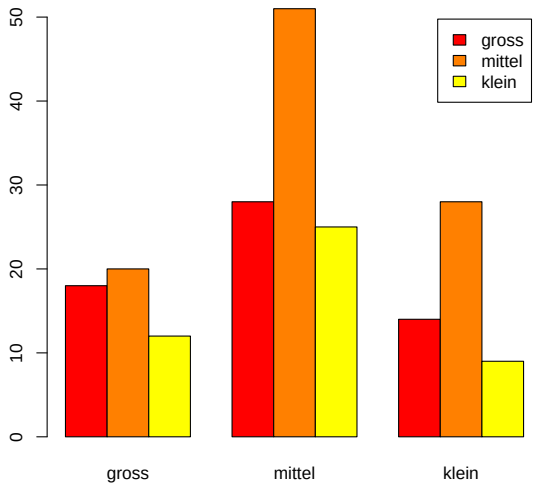
Beispiel 3.1 (Körpergröße bei 205 Ehepaaren)

Hier haben wir keine Rohdaten und erzeugen die Tabelle direkt:

```
R> absH <- c(18, 28, 14, 20, 51, 28, 12, 25,
+     9)
R> tab <- matrix(absH, nr = 3, byrow = TRUE)
R> tab

     [,1] [,2] [,3]
[1,]   18   28   14
[2,]   20   51   28
[3,]   12   25    9

R> dimnames(tab) <- list(Ehemann = c("gross",
+     "mittel", "klein"), Ehefrau = c("gross",
+     "mittel", "klein"))
R> tab

        Ehefrau
Ehemann  gross mittel klein
  gross     18     28    14
  mittel    20     51    28
  klein     12     25     9
```

Grafische Darstellung der gemeinsamen kategorialen Information mittels eines gruppierten Balkendiagramms:

```
R> barplot(tab, beside = TRUE, legend = TRUE)
```



Gestapelte Balkendiagramme erzeugt man direkt mit `barplot`, wenn deren Argument eine Matrix (also beispielsweise via `prop.table` erzeugte bedingte Information) ist: die Spalten der Matrix werden dann übereinander gestapelt. Wenn man stattdessen die Zeilen haben möchte, muss man Zeilen und Spalten und vertauschen, also die Matrix transponieren: dazu dient die Funktion `t`.

Beispiel 3.2 (Einschätzung des eigenes Gewichtes bei Teenagern)

Wir erzeugen die Tabelle selbst:

```
R> absH <- c(42, 65, 224, 121, 12, 212)
R> tab <- matrix(absH, nr = 2, byrow = TRUE)
R> tab
```

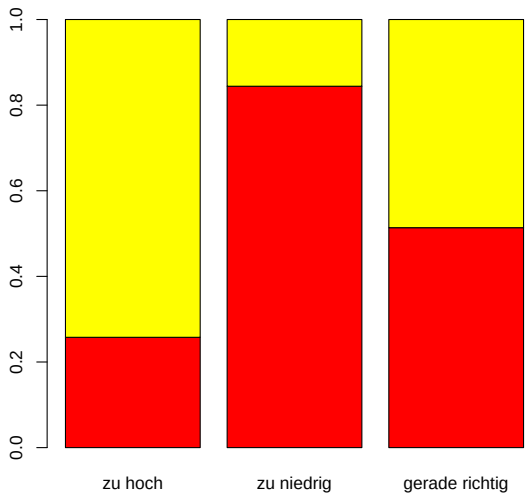
```
     [,1] [,2] [,3]
[1,]   42   65  224
[2,]  121   12  212
```

```
R> dimnames(tab) <- list(Geschlecht = c("maennlich",
+     "weiblich"), Gewicht = c("zu hoch", "zu niedrig",
+     "gerade richtig"))
R> tab
```

```
           Gewicht
Geschlecht  zu hoch zu niedrig gerade richtig
  maennlich      42         65            224
  weiblich      121         12            212
```

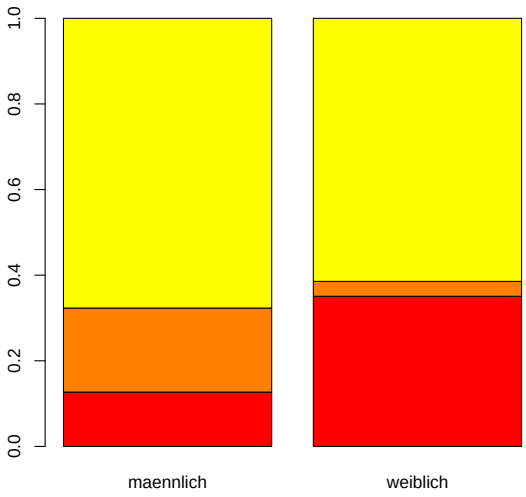
Grafische Darstellung der bedingten Information nach Gewicht (hier werden die Spaltenhäufigkeiten gebildet):

```
R> barplot(prop.table(tab, 2))
```



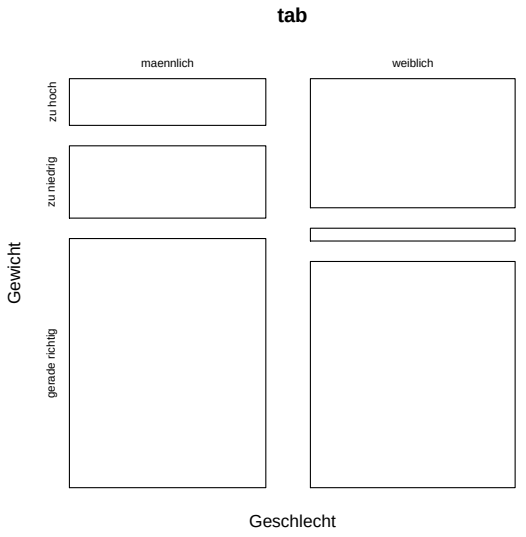
Grafische Darstellung der bedingten Information nach Geschlecht (hier werden die Zeilenhäufigkeiten gebildet, zum Übereinanderstapeln muss man noch Zeilen und Spalten vertauschen):

```
R> barplot(t(prop.table(tab, 1)))
```



Eine mit dieser Darstellungsmöglichkeit eng verwandte Form der Visualisierung sind Mosaicplots, eine flächenproportionale Darstellung der Zellhäufigkeiten von Kontingenztabellen:

```
R> mosaicplot(tab)
```



3.3 Ist die Verteilung von Häufigkeiten in verschiedenen Gruppen gleich?

Für den χ^2-Homogenitätstest verwendet man die Funktion `chisq.test`, wobei man als Argumente entweder die beiden Vektoren mit den Werten der kategorialen Variablen oder deren Kreuztabelle nimmt. (Ein Bilden von bedingten Häufigkeiten ist nicht erforderlich.)

Beispiel 3.3 (Präsentatorenwerbung und normale Werbung)

Einlesen der Daten in bekannter Art und Weise:

```
R> CMMRCIAL <- read.table("cmmrcial.tab", header = TRUE)
R> attach(CMMRCIAL)
R> chisq.test(CEREAL, GROUP)

        Pearson's Chi-squared test

data:  CEREAL and GROUP
X-squared = 1.8333, df = 3, p-value = 0.6077
```

Der p-Wert von rund 0.6 zeigt, dass die beobachteten Unterschiede in den bedingten Verteilungen dem Zufall zuzuschreiben sind.

Die beobachteten und (unter der Nullhypothese der Homogenität) erwarteten Häufigkeiten bekommen wir wieder als Komponenten des Wertes des Tests:

```
R> testErg <- chisq.test(CEREAL, GROUP)
R> list(beobachtet = testErg$observed, erwartet = testErg$expected)

$beobachtet
      GROUP
CEREAL NW SW
    BB 31 33
    CC 37 44
    FL 26 19
    KH 27 25

$erwartet
      GROUP
CEREAL   NW   SW
    BB 32.0 32.0
    CC 40.5 40.5
    FL 22.5 22.5
    KH 26.0 26.0
```

(und könnten diese so auch mittels gruppierter Balkendiagramme grafisch darstellen).

3.4 Sind zwei kategoriale Merkmale voneinander unabhängig?

Den χ^2-Unabhängigkeitstest führt man identisch zum Test auf Homogenität durch.

Beispiel 3.4 (Erwartete Verkäufe im In- und Ausland)

Einlesen der Daten in bekannter Art und Weise aus einer „.csv" Datei:

```
R> SALES <- read.csv(file = "expectedsales.csv")
R> SALES

  COUNT DOMESTIC  FOREIGN
1    78 Increase Increase
2    88 Increase Decrease
3    49 Decrease Increase
4    37 Decrease Decrease
```

Hier haben wir keine Rohdaten, sondern bereits vortabellierte Werte, deren Häufigkeiten in der Variablen `COUNT` abgespeichert sind. Um hieraus die Kontingenztafel zu bauen, verwenden wir wieder den Befehl `xtabs`

```
R> tab <- xtabs(COUNT ~ FOREIGN + DOMESTIC, data = SALES)
R> tab

          DOMESTIC
FOREIGN    Decrease Increase
  Decrease 37       88
  Increase 49       78
```

wobei nun auf der linken Seite in der Formel die Häufigkeiten stehen, die zur Erzeugung der Kreuztabelle benutzt werden sollen. Auf die so erzeugte Tabelle koennen wir wiederum den Befehl `chisq.test` anwenden, um die Abhängigkeit der beiden Variablen zu testen:

```
R> chisq.test(tab)

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

data:  tab
X-squared = 1.8792, df = 1, p-value = 0.1704
```

Aus der Zusammenfassung können wir ablesen, daß bei der Berechnung der Teststatistik eine Endlichkeitskorrektur verwendet wurde. Um die Statistik in herrkömmlicher Weise zu berechnen, muß diese abgeschaltet werden.

```
R> chisq.test(tab, correct = FALSE)

        Pearson's Chi-squared test

data:  tab
X-squared = 2.2611, df = 1, p-value = 0.1327
```

In beiden Fällen weist der große p-Wert darauf hin, daß kein Zusammenhang zwischen dem erwarteten Zuwachs an Verkäufen im Ausland (`FOREIGN`) und im Inland (`DOMESTIC`) besteht.

3.5 Wie kann man den Zusammenhang zweier kategorialer Merkmale beschreiben?

Wenn man einen Zusammenhang zwischen zwei kategorialen Merkmalen nachweisen kann, kann man diesen durch das Chancenverhältnis („Odds Ratio“) numerisch quantifizieren und durch Einfärbung („Shading“) von Mosaicplots oder durch Assoziationsplots visualisieren.

Beispiel 3.5 (Produktvermarktung)

Einlesen der Daten in bekannter Art und Weise aus einer ‚.sav‘ SPSS-Datei:

```
R> PROMOT <- as.data.frame(read.spss(file = "promot.sav"))
R> PROMOT

  COUNT TRY         PROMOT VAR00004 VAR00006
1    58 yes         dinner 2.741906 1.008653
2    23  no         dinner       NA       NA
3    47 yes representative       NA       NA
4    38  no representative       NA       NA
```

Hier haben wir keine Rohdaten, sondern bereits vortabellierte Werte, deren Häufigkeiten in der Variablen `COUNT` abgespeichert sind. Um hieraus die Kontingenztafel zu bauen, verwenden wir wieder den Befehl `xtabs`

```
R> tab <- xtabs(COUNT ~ TRY + PROMOT, data = PROMOT)
R> tab
```

```
     PROMOT
TRY   representative dinner
  no  38             23
  yes 47             58
```

wobei nun auf der linken Seite in der Formel die Häufigkeiten stehen, die zur Erzeugung der Kreuztabelle benutzt werden sollen. Auf die so erzeugte Tabelle koennen wir wiederum den Befehl `chisq.test` anwenden, um die Abhängigkeit der beiden Variablen zu testen:

```
R> chisq.test(tab)

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

data:  tab
X-squared = 4.0715, df = 1, p-value = 0.04361
```

Aus der Ausgabe dieses Aufrufs können wir ablesen, daß bei der Berechnung der Teststatistik eine Endlichkeitskorrektur verwendet wurde. Möchte man die Statistik in herrkömmlicher Weise zu berechnen, kann diese abgeschaltet werden.

```
R> chisq.test(tab, correct = FALSE)

        Pearson's Chi-squared test

data:  tab
X-squared = 4.7473, df = 1, p-value = 0.02934
```

In beiden Fällen weist der p-Wert von kleiner 5% darauf hin, daß ein Zusammenhang zwischen der Art der Produktvermarktung (`PROMOT`) und der Wahl des Produkts (`TRY`) besteht. Um nun den Zusammenhang numerisch genauer zu quantifizieren, können wir den Odds Ratio berechnen

```
R> or1 <- (38 * 58)/(47 * 23)
R> or1

[1] 2.038853
```

woraus wir ablesen können, daß die Chancen das neue Produkt *nicht* ausprobieren rund doppelt so gross sind, wenn ein Vertreter geschickt wurde anstatt einer Einladung zu einer Dinner-Party. Durch Bildung des Kehrwertes des Odds Ratios bzw. durch Vertauschung der Zeilen (oder der Spalten) vor Berechnung des Odds Ratios

```
R> tab[2:1, ]

     PROMOT
TRY   representative dinner
  yes             47     58
  no              38     23

R> or2 <- (58 * 38)/(23 * 47)
R> or2

[1] 2.038853

R> 1/or1

[1] 0.4904719
```

sehen wir völlig äquivalent, dass die Chancen das neue Produkt auszuprobieren nur etwa halb so groß sind, wenn ein Vertreter geschickt wurde anstatt einer Einladung zu einer Dinner-Party.

Eine allgemeinere Methode, um den Zusammenhang zweier kategorialer Merkmale zu modellieren und zu testen, sind Binomial- bzw. Multinomialmodelle. Diese sind insbesonders angemessen, wenn eine Variable die abhängige Variable ist, wie `TRY` in diesem Beispiel. Das entsprechende Binomialmodell, das das Ausprobieren des Produktes `TRY` durch die Vermarktung `PROMOT` zu erklären versucht, kann durch die Funktion `glm` angepaßt werden, wobei das Argument `family` auf `binomial` gesetzt wird. Das erste Argument bei dem Funktionsaufruf ist wieder eine Formel, wobei die `~` wieder als "wird erklärt durch" gelesen werden kann. Das Argument `weights` spezifiziert die Zellhäufigkeiten, da wir hier wieder vortabellierte Werte vorliegen haben und `data` den Datensatz, in dem die verwendeten Daten zu finden sind.

```
R> glm(TRY ~ PROMOT, weights = COUNT, data = PROMOT,
+     family = binomial)

Call:  glm(formula = TRY ~ PROMOT, family = binomial, data = PROMOT,      weights = COUNT)

Coefficients:
 (Intercept)  PROMOTdinner
      0.2126        0.7124

Degrees of Freedom: 3 Total (i.e. Null);  2 Residual
Null Deviance:            218.3
Residual Deviance: 213.5         AIC: 217.5
```

Als Ergebnis bekommen wir die geschätzten Koeffizienten dieses Modells. Um eine ausführlichere Zusammenfassung zu bekommen, können wir wieder die Funktion `summary` verwenden:

```
R> fm <- glm(TRY ~ PROMOT, weights = COUNT, data = PROMOT,
+     family = binomial)
R> summary(fm)

Call:
glm(formula = TRY ~ PROMOT, family = binomial, data = PROMOT,
    weights = COUNT)

Deviance Residuals:
     1       2       3       4
 6.225  -7.610   7.463  -7.822

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    0.2126     0.2182   0.974   0.3299
PROMOTdinner   0.7124     0.3291   2.165   0.0304 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 218.32  on 3  degrees of freedom
Residual deviance: 213.54  on 2  degrees of freedom
AIC: 217.54

Number of Fisher Scoring iterations: 4
```

Daraus läßt sich ablesen, daß die Vermarktung (`PROMOT`) einen signifikanten Einfluß auf die Wahrscheinlichkeit das Produkt auszuprobieren (`TRY`) hat, da der zugehörgie p-Wert kleiner als 5% ist. Möchte man mit einem χ^2-Test testen, ob die Variable `PROMOT` einen Einfluß hat, kann man dies mit der Funktion `anova` tun

```
R> anova(fm, test = "Chisq")

Analysis of Deviance Table

Model: binomial, link: logit
```

```
Response: TRY

Terms added sequentially (first to last)


       Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                       3    218.322
PROMOT  1    4.785         2    213.537     0.029
```

die ebenfalls ein signifikantes Ergebnis liefert. Aus dem angepaßten Modell kann man die Koeffizienten extrahieren und durch Anwendung der Exponentialfunktion darauf auch wieder den Odds Ratio berechnen

```
R> coef(fm)

 (Intercept) PROMOTdinner
   0.2125614    0.7123874

R> exp(coef(fm)[2])

PROMOTdinner
    2.038853
```

was das Ergebnis der Berechnungen von oben liefert. Wozu braucht man also überhaupt das Binomialmodell, wenn es doch sowieso im Wesentlichen dieselben Ergebnisse liefert wie die Berechnungen des Odds Ratio und des χ^2-Tests auf Basis der 2×2 Kontingenztafel? Der Grund ist, daß das Binomialmodell auch auf gr"oßere Kontingenztafeln angewendet werden kann, solange diese genau eine abhängige Variable mit zwei Ausprägungen hat. Bevor wir uns ein solches Beispiel anschauen, räumen wir noch auf.

```
R> remove(PROMOT, tab, or1, or2, fm)
```

Beispiel 3.6 (Autokauf)

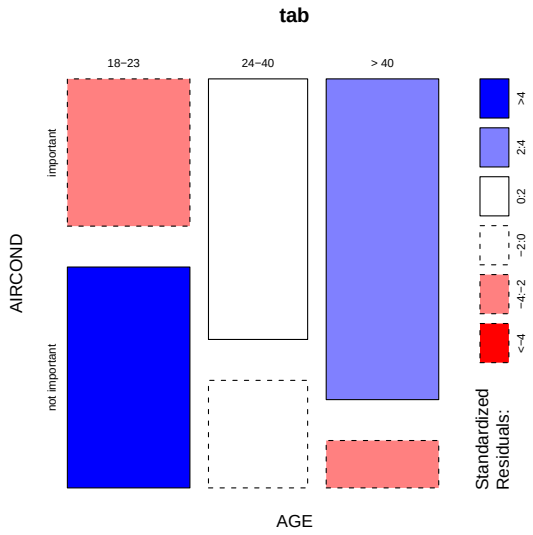
Zunächst lesen wir den Datensatz auf bekannte Art und Weise ein und bilden die Kontigenztafeln, die die Wichtigkeit einer Klimaanlage beim Autokauf über verschiedene Altersgruppen zeigt.

```
R> CAR <- read.csv(file = "carfeatures.csv")
R> tab <- xtabs(COUNTS ~ AGE + AIRCOND, data = CAR)
R> tab
```

```
          AIRCOND
AGE        important not important
  18-23    44        66
  24-40    63        26
  > 40     88        13
```

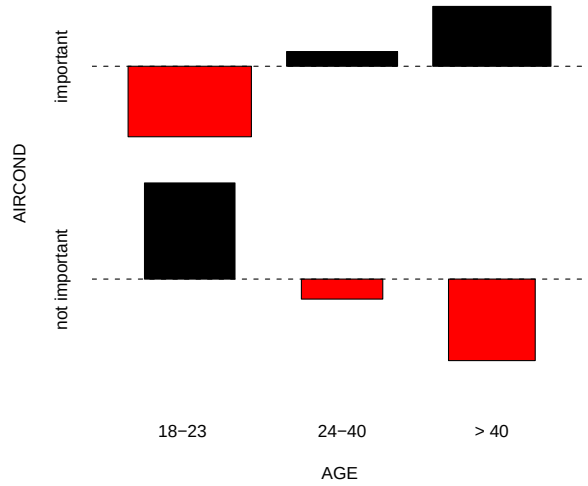
Um uns zunächst einen grafischen Eindruck zu verschaffen, visualisieren wir die Kontingenztafel mit Hilfe eines Mosaicplots mit „Shading“.

```
R> mosaicplot(tab, shade = TRUE)
```



Dabei werden die Zellen des Mosaicplots gemäss der zugehörigen Summanden der χ^2-Statistik, den sogenannten Pearson-Residuen, eingefärbt. Dies läßt sich sehr leicht interpretieren: blaue Zellen werden häufiger und rote Zellen weniger häufig beobachtet als unter Unabhängigkeit erwartet würde. Anstatt diese Pearson-Residuen als Grundlage fuer eine Einfärbung zu verwenden, kann man sie auch selbst in einem Assoziationsplot visualisieren.

```
R> assocplot(tab)
```



Für den Autodatensatz heißt das also, daß weniger 18-23jährige gibt, denen eine Klimaanlage beim Autokauf wichtig ist, (bzw. mehr denen sie unwichtig ist) und mehr über 40jährige denen sie wichtig ist (bzw. weniger denen sie unwichtig ist) als erwartet würde, wenn die Wichtigkeit der Klimaanlage vom Alter unabhängig wäre. Dies deutet also auf einen signifikanten Zusammenhang hin, was durch Anpassung des entsprechenden Binomialmodells und Durchführung des entsprechenden Tests auch bestätigt werden kann.

```
R> fm <- glm(AIRCOND ~ AGE, weights = COUNTS,
+     data = CAR, family = binomial)
R> fm

Call:  glm(formula = AIRCOND ~ AGE, family = binomial, data = CAR, weights = COUNTS)

Coefficients:
(Intercept)     AGE24-40      AGE> 40
     0.4055      -1.2905      -2.3179

Degrees of Freedom: 17 Total (i.e. Null);  15 Residual
Null Deviance:            388.5
Residual Deviance: 333.1         AIC: 339.1

R> anova(fm, test = "Chisq")
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: AIRCOND

Terms added sequentially (first to last)


     Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    17     388.47
AGE   2    55.33        15     333.14 9.667e-13
```

Auch in solchen Kontingenztafeln kann man nun Odds Ratios fuer 2×2 Teiltabellen berechnen. Beispielsweise entspricht der Odds Ratio

```
R> or <- exp(coef(fm)[3])
R> 1/or

 AGE> 40
10.15385
```

der Interpretation, daß die Chance, daß eine Klimaanlage wichtig ist, in der Gruppe der über 40jährigen rund 10mal so hoch ist wie in der Referenzgruppe der 18-23jährigen.

Zum Schluß räumen wir wieder zusammen:

```
R> remove(CAR, tab, fm, or)
```

Kapitel 4

Ein metrisches Merkmal

4.1 Wie kann man ein metrisches Merkmal numerisch beschreiben?

Häufigkeitstabellen erzeugt man mit `table`.

Das arithmetische Mittel und den Median berechnet man mit `mean` und `median`.

Für die Berechnung der Quantile gibt es `quantile`.

Varianz, Standardabweichung und mittlere absolute Abweichung berechnet man mit den Funktionen `var`, `sd` und `mad`.

Die Funktion `range` liefert einen Vektor mit dem Minimum und dem Maximum der Daten; durch Differenzbildung mit `diff` erhält man daraus die Spannweite.

Den Interquartilsabstand berechnet man mit `IQR`.

4.2 Wie kann man ein metrisches Merkmal grafisch beschreiben?

Stem-and-leaf Plots erzeugt man mit `stem`.

Histogramme erzeugt man mit `hist`. Im Falle gleich breiter Klassen werden standardmäßig absolute Klassenhäufigkeiten dargestellt (sonst relative): will man stets relative Häufigkeiten verwenden, so kann man dies erreichen indem man das zusätzliche Argument `prob = TRUE` verwendet. Bei der Berechnung der Klasseneinteilung werden standardmäßig zeitgemäße automatische Verfahren verwendet, man kann die Anzahl der Intervalle oder auch die Teilungspunkte mit dem Argument `breaks` festlegen.

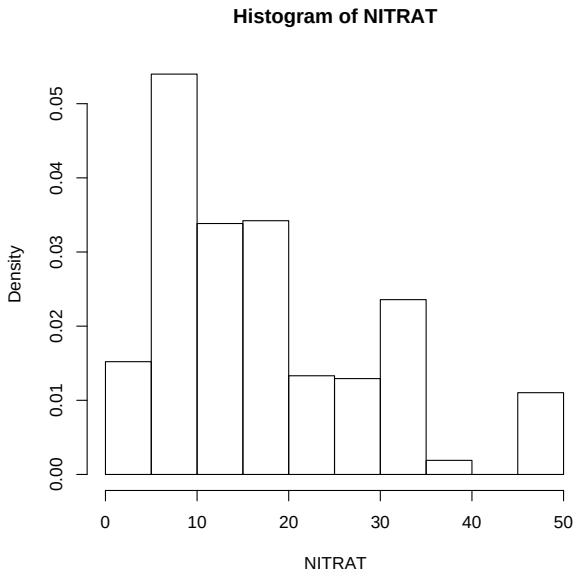
Beispiel 4.1 (Nitratbelastung im niederösterreichischen Trinkwasser)

Einlesen der Daten:

```
R> NOEWASSER <- read.table("noewasser.tab", header = TRUE)
R> dim(NOEWASSER)

[1] 526  12

R> names(NOEWASSER)

 [1] "GEMEINDE" "ENTNAHME" "DATUM"    "PHWERT"
 [5] "GESAMTH." "NITRAT"   "EISEN"    "MANGAN"
 [9] "CHLORID"  "SULFAT"   "ORT"      "GEWICHT"
```

Histogramm mittels `hist`:

```
R> attach(NOEWASSER)
R> hist(NITRAT, prob = TRUE)
```



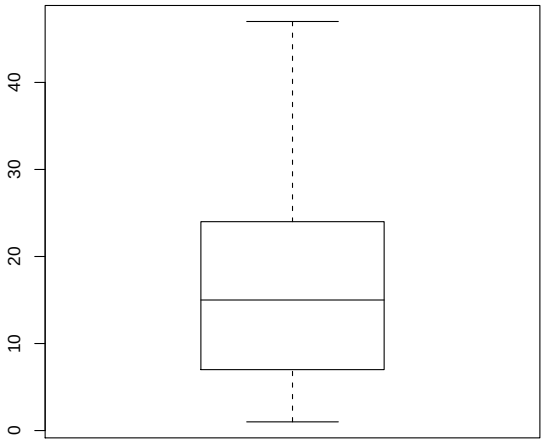
Durch Angabe verschiedener Werte für `breaks` kann man die „Auflösung“ der Darstellung verändern.

Boxplots erzeugt man mit `boxplot`. Standardmäßig werden die „Schnurrbarthaare“ (englisch: whiskers, also die Linien von der Box weg in Richtung Minimum und Maximum) nur bis zum 1.5-fachen Abstand von der Box gezeichnet (auch, um extremere Werte als „Ausreißer“ erkennen zu können); mit dem Argument `range = 0` erreicht man, dass die Schnurrbarthaare vom Minimum bis zum Maximum gehen.

Beispiel 4.2 (Nitratbelastung im niederösterreichischen Trinkwasser)

Die Daten haben wir bereits weiter oben eingelesen. Boxplot mittels `boxplot`:

```
R> boxplot(NITRAT)
```



Zusammenräumen:

```
R> detach("NOEWASSER")
R> remove(NOEWASSER)
```

4.3 Haben die Daten eine bestimmte Verteilungsform?

QQ-Plots der Quantile eines Datenvektors gegen die Quantile der Normalverteilung („normal QQ plot“) erzeugt man mit `qqnorm`; mit `qqline` kann man eine Gerade hinzufügen, die durch das untere und obere Quartil geht.

Beispiel 4.3 (PS bei einer Stichprobe von 406 PKWs (U.S.A.))

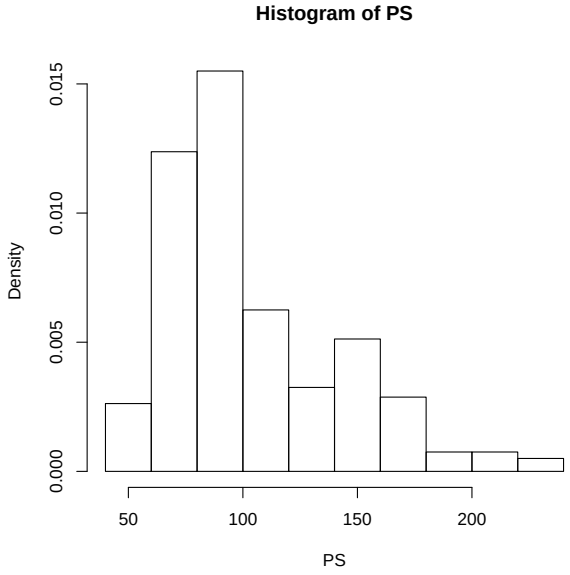
Einlesen der Daten:

```
R> AUTOS <- read.table("autos.tab", header = TRUE)
R> dim(AUTOS)

[1] 406  10
```

```
R> names(AUTOS)

 [1] "MPG"      "HUBRAUM"  "PS"       "GEWICHT"
 [5] "BESCHLEU" "BAUJAHR"  "LAND"     "ZYLINDER"
 [9] "FILTER.." "GALPMIL"
```

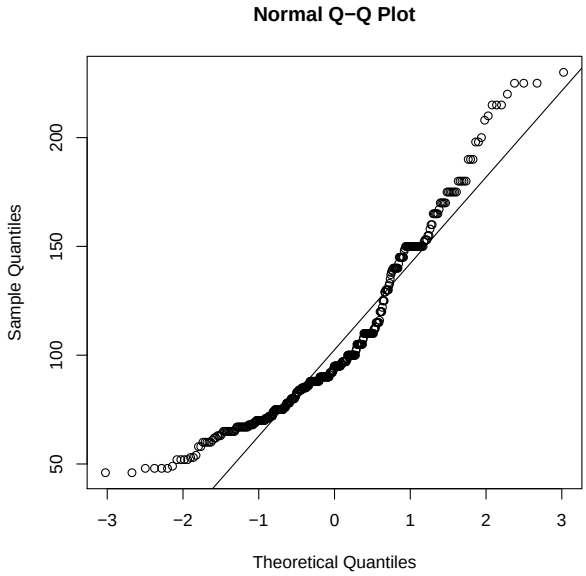
Ein Histogramm von PS

```
R> attach(AUTOS)
R> hist(PS, prob = TRUE)
```



lässt bereits erkennen, dass die Daten nicht gut durch eine Normalverteilung beschreibbar sind. Auch im QQ-Plot

```
R> qqnorm(PS)
R> qqline(PS)
```



sehen wir, dass die Punkte „nicht wirklich“ auf einer Geraden liegen.

Zusammenräumen:

```
R> detach("AUTOS")
R> remove(AUTOS)
```

4.4 In welchem Bereich kann man einen Mittelwert in der Grundgesamtheit erwarten?

Mit der Funktion `t.test` kann man Inferenz über Mittelwerte (bei unbekannter Varianz in der Grundgesamtheit) treiben: sie liefert sowohl Konfidenzintervalle als auch Tests von Hypothesen.

(Wir gehen davon aus, dass die Varianz in der Grundgesamtheit de facto immer unbekannt ist.)

Beispiel 4.4 (National Patent Development Corporaton (NPDC), CARIDEX: Anzahl behandelter Zähne pro Woche)

Einlesen der Daten:

```
R> CARIDEX <- read.table("caridex.tab", header = TRUE)
R> dim(CARIDEX)
```

```
[1] 400   1
```

```
R> names(CARIDEX)
```

```
[1] "CAVITIES"
```

Wir haben also 400 Beobachtungen einer Variable: `CAVITIES` (Anzahl von Behandlungen pro Woche).

Ein 95%-Konfidenzintervall für die zu erwartende Anzahl von Behandlungen in der Grundgesamtheit bekommen wir via

```
R> attach(CARIDEX)
R> t.test(CAVITIES)

        One Sample t-test

data:  CAVITIES
t = 45.5152, df = 399, p-value = < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 3.841581 4.188419
sample estimates:
mean of x
    4.015
```

als $(3.842, 4.188)$.

Um ein Konfidenzintervall mit 99%-iger Sicherheit zu erhalten, verwenden wir

```
R> t.test(CAVITIES, conf.level = 0.99)$conf.int
```

```
[1] 3.786689 4.243311
attr(,"conf.level")
[1] 0.99
```

Zusammenräumen:

```
R> detach("CARIDEX")
R> remove(CARIDEX)
```

4.5 Ist ein Mittelwert in der Grundgesamtheit anders als in einer bestimmten Vorgabe?

Hypothesen über den Mittelwert in der Grundgesamtheit (bei unbekannter Varianz in der Grundgesamtheit) überprüft man mit `t.test`. Den Wert μ_0 des Mittelwertes unter der Nullhypothese spezifiert man dabei mittels des Argumentes `mu`; das Argument `alternative` erlaubt es die Alternativen $\mu < \mu_0$ (`"less"`), $\mu \neq \mu_0$ (`"two.sided"`) und $\mu > \mu_0$ (`"greater"`) zu wählen.

(Wir gehen davon aus dass, die Varianz in der Grundgesamtheit de facto immer unbekannt ist.)

Beispiel 4.5 (Diät: durchschnittliche Gewichtsabnahme von mehr als 5kg?)

Einlesen der Daten:

```
R> DIET <- read.table("diet.tab", header = TRUE)
R> dim(DIET)

[1] 20  1

R> names(DIET)

[1] "GEWICHT"
```

Wir haben also 20 Beobachtungen einer Variable: `GEWICHT` (Gewichtsreduktion nach 2-wöchiger Diät).

Um zu testen, ob man mit der Diät eine durchschnittliche Gewichtsabnahme von mehr als 5 kg erzielt, müssen wir die Nullhypothese $\mu = 5$ gegen die Alternative $\mu < 5$ testen (wobei μ der Mittelwert der Gewichtsabnahme in der Grundgesamtheit ist).

```
R> attach(DIET)
R> t.test(GEWICHT, mu = 5, alternative = "less")

        One Sample t-test

data:  GEWICHT
t = -0.7669, df = 19, p-value = 0.2263
alternative hypothesis: true mean is less than 5
95 percent confidence interval:
     -Inf 5.344555
sample estimates:
mean of x
 4.725377
```

Angesichts des p-Wertes von 0.23 können wir die Nullhypothese nicht verwerfen: in der Stichprobe liegt die mittlere Gewichtsabnahme zwar bei 4.73 und somit unter 5 kg, aber eben nicht „signifikant genug“.

Zusammenräumen:

```
R> detach("DIET")
R> remove(DIET)
```

Kapitel 5

Mehrere metrische Merkmale

5.1 Wie kann man zwei metrische Merkmale grafisch beschreiben?

Ein Streudiagram von zwei metrischen Merkmalen erzeugt man mit der Funktion `plot` mit den Beobachtungen der Variablen als Argumente. Die erste Variable wird dann auf der x-, die zweite auf der y-Achse aufgetragen.

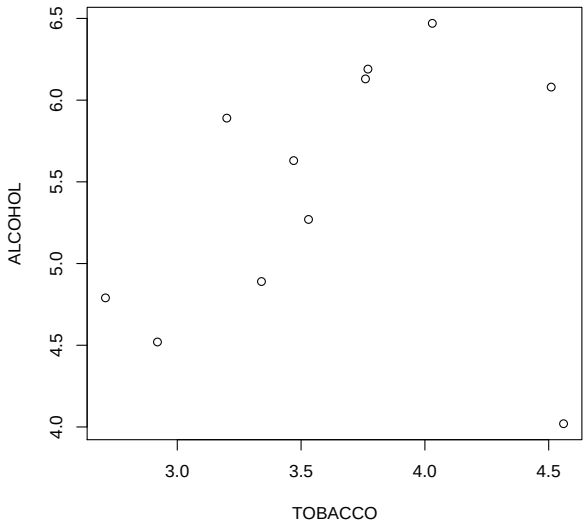
Beispiel 5.1 (Ausgaben für Alkohol und Tabak)

Daten einlesen:

```
R> ALCTOBAC <- read.table("alctobac.tab", header = TRUE)
R> dim(ALCTOBAC)

[1] 11  3

R> names(ALCTOBAC)

[1] "REGION"  "ALCOHOL" "TOBACCO"
```

Ein Streudiagramm mit `plot`:

```
R> attach(ALCTOBAC)
R> plot(TOBACCO, ALCOHOL)
```



5.2 Wie kann man zwei metrische Merkmale numerisch beschreiben?

Den Korrelationskoeffizienten (von K. Pearson) und die Kovarianz berechnet man mit den Funktionen `cor` und `cov`.

Beispiel 5.2 (Ausgaben für Alkohol und Tabak)

Die Daten haben wir schon weiter oben eingelesen.

```
R> cor(TOBACCO, ALCOHOL)

[1] 0.2235721
```

ergibt einen relativ niedrigen Wert von 0.22. Allerdings ist im Streudiagrammm ein Punkt als „Ausreißer“ zu erkennen, den wir mittels

```
R> outlierIndex <- which((ALCOHOL < 5) & (TOBACCO >
+     4))
R> outlierIndex

[1] 11
```

```
R> ALCTOBAC[outlierIndex, ]

            REGION ALCOHOL TOBACCO
11 NorthernIreland    4.02    4.56
```

unschwer als Nordirland erkennen können.

Ohne Nordirland erhalten wir

```
R> cor(TOBACCO[-11], ALCOHOL[-11])

[1] 0.7842873
```

5.3 Wie stark ist der Zusammenhang zwischen zwei metrischen Merkmalen?

Für Inferenz über den Korrelationskoeffizienten verwendet man `cor.test` (mit dem Argument `method` kann man auch die Verwendung von Rangkorrelationskoeffizienten anstelle des Pearson'schen Korrelationskoeffizientens erreichen).

Beispiel 5.3 (Ausgaben für Alkohol und Tabak)

Die Daten haben wir schon weiter oben eingelesen.

```
R> cor.test(TOBACCO, ALCOHOL)

        Pearson's product-moment correlation

data:  TOBACCO and ALCOHOL
t = 0.6881, df = 9, p-value = 0.5087
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.4345878  0.7260700
sample estimates:
      cor
0.2235721
```

liefert ein 95%-Konfidenzintervall für den Korrelationskoeffizienten und p-Wert von 0.51, der uns nicht erlaubt die Nullhypothese, dass kein Zusammenhang zwischen den Ausgaben für Tabakwaren und alkoholischen Getränken besteht, zu verwerfen.

Wenn wir dagegen Nordirland exkludieren, erhalten wir

```
R> cor.test(TOBACCO[-11], ALCOHOL[-11])
```

```
        Pearson's product-moment correlation

data:  TOBACCO[-11] and ALCOHOL[-11]
t = 3.5756, df = 8, p-value = 0.007234
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.3055382 0.9465163
sample estimates:
      cor
0.7842873
```

und können somit aufgrund des p-Wertes von 0.007 auf einen signifikanten Zusammenhang schließen.

5.4 Welche Form hat der Zusammenhang zwischen zwei metrischen Merkmalen?

Lineare Regressionsmodelle baut man mit der Funktion `lm`, der man als (erstes) Argument eine Modellformel übergibt, in der, durch eine Tilde getrennt, links die abhängige und rechts die erklärende Variable steht.

Beispiel 5.4 (Gebrauchtwagenpreise (U.S.A.))

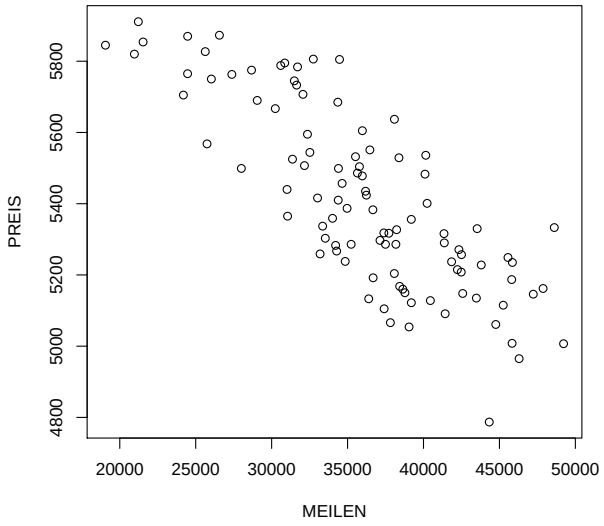
Einlesen der Daten:

```
R> SECONDHANDCAR <- read.table("secondhandcar.tab",
+     header = TRUE)
R> dim(SECONDHANDCAR)

[1] 100  11

R> names(SECONDHANDCAR)

 [1] "PREIS"   "MEILEN"  "COLOR"   "I1"      "I2"
 [6] "P"       "O"       "RES.1"   "PRE.1"   "PRE.2"
[11] "SERVICE"
```

Es handelt sich also um einen Datensatz von 100 Beobachtungen von insgesamt 11 Variablen. Wir wollen untersuchen, wie sich der Gebrauchtwagenpreis (`PREIS`) in Abhängigkeit vom „Kilometerstand“ (`MEILEN`) verhält.

Das Streudiagramm

```
R> attach(SECONDHANDCAR)
R> plot(MEILEN, PREIS)
```



suggeriert einen ausgeprägten negativen linearen Zusammenhang. Die Koeffizienten der Regressionsgerade erhalten wir mit

```
R> lm(PREIS ~ MEILEN)

Call:
lm(formula = PREIS ~ MEILEN)

Coefficients:
(Intercept)       MEILEN
 6533.38303     -0.03116
```

(lies: „lineares Modell von `PREIS` nach `MEILEN`“). Die Gleichung der Regressionsgerade ergibt sich zu

$$\text{PREIS} = 6533.38 - 0.031 * \text{MEILEN};$$

im linearen Modell reduziert sich also der Preis pro zusätzlich gefahrener Meile um USD 0.031.

Wenn wir das Ergebnis des Schätzens des linearen Modells (Berechnung der Regressionkoeffizienten) einem R Objekt zuweisen, so können wir erkennen, dass dieses von einem

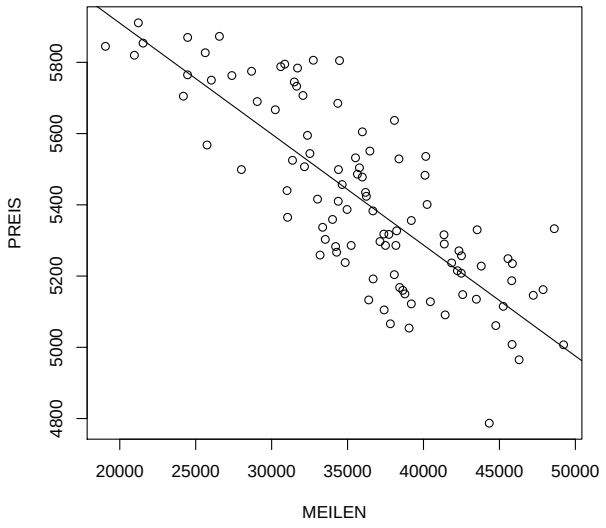
speziellen Typ (eine Liste der Klasse `"lm"`) ist und zusätzliche Information in weiteren Komponenten enthält:

```
R> shcLM <- lm(PREIS ~ MEILEN)
R> class(shcLM)

[1] "lm"

R> names(shcLM)

 [1] "coefficients"  "residuals"     "effects"
 [4] "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "xlevels"
[10] "call"          "terms"         "model"
```

Mit solchen Modellobjekten kann man eine Reihe weiterer wichtiger Operationen ausführen.

Zunächst können wir die Regressionsgerade mittels `abline` in das Streudiagramm einzeichnen:

```
R> plot(MEILEN, PREIS)
R> abline(shcLM)
```



Weiters können wir für gegebene Meilenstände die Preise vorhersagen (durch „Einsetzen in die Regressiongerade“):

```
R> predict(shcLM, list(MEILEN = 40000))

[1] 5287.073
```

Dann liefert `summary` Information über die Modellgüte und p-Werte für die üblichen Hypothesentests im linearen Modell, insbesondere für den Test der Hypothese, dass kein linearer Zusammenhang besteht ($\beta = 0$):

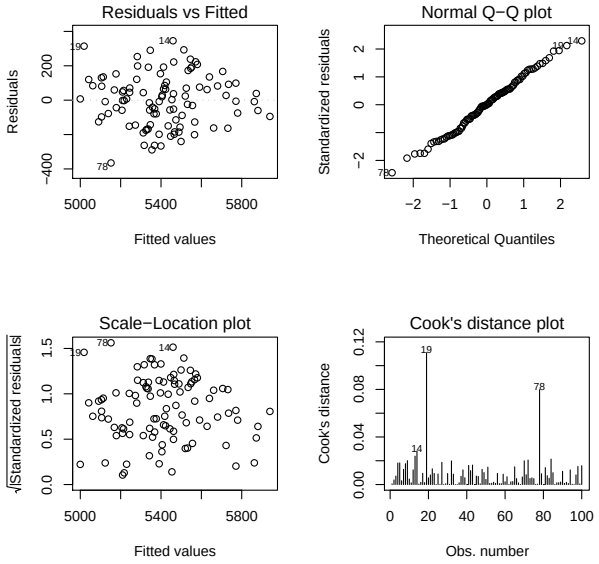
```
R> summary(shcLM)

Call:
lm(formula = PREIS ~ MEILEN)

Residuals:
      Min        1Q    Median        3Q       Max
-365.1605 -117.5073    0.6528   93.8729  345.6242

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.533e+03  8.451e+01   77.31   <2e-16 ***
MEILEN      -3.116e-02  2.309e-03  -13.49   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 151.6 on 98 degrees of freedom
Multiple R-Squared: 0.6501,        Adjusted R-squared: 0.6466
F-statistic: 182.1 on 1 and 98 DF,  p-value: < 2.2e-16
```

Aus den letzten Zeilen erkennen wir ein lineares Bestimmtheitsmaß von $R^2 = 0.65$, es werden also rund 65 Prozent der Varianz von `PREIS` durch das einfache Regressionsmodell erklärt.

Den p-Wert für den Test der Hypothese, dass der entsprechende Regressionkoeffizient 0 ist, findet man in der entsprechenden Zeile und letzten Spalte (`Pr(>|t|)`) der Subtabelle `Coefficients`. Der p-Wert ist jedenfalls < 0.001; es besteht somit ein deutlich signifikanter linearer Zusammenhang.

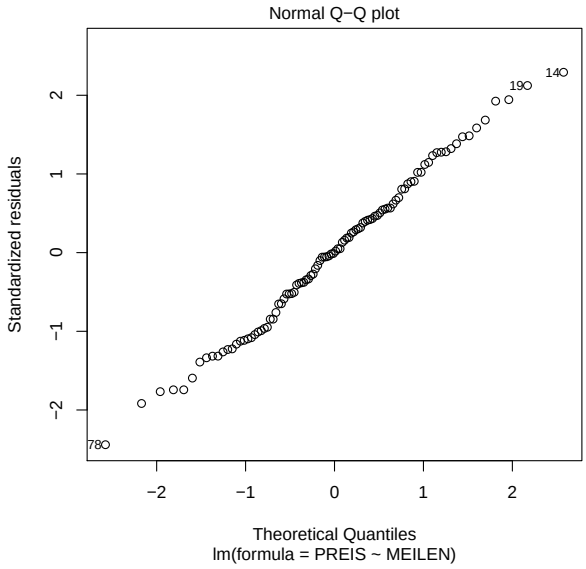
Schließlich kann man mittels `plot` diagnostische Plots zur grafischen Überprüfung des Modells erzeugen. Standardmäßig liefert dies der Reihe nach 4 Grafiken (Residuen-Plot, QQ-Plot der Residuen, und zwei weitere), die hier der Einfachheit halber in einer Grafik zusammengefasst werden:

```
R> opar <- par(mfrow = c(2, 2))
R> plot(shcLM)
R> par(opar)
```

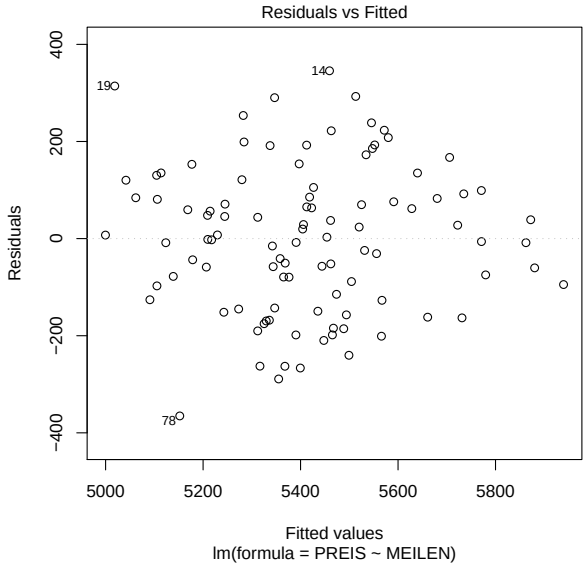


Durch Angabe des Argumentes `which` kann man auch einzelne Plots selektieren (mit `which = 1` den Residuen-Plot und mit `which = 2` den QQ-Plot der Residuen):

```
R> plot(shcLM, which = 2)
```




```
R> plot(shcLM, which = 1)
```



Zum Bauen von multiplen linearen Regressionsmodelle verwendet man wieder `lm` und schreibt auf der rechten Seite der Modellformel alle erklärenden Variablen, verknüpft durch ‚+‘:

```
R> shcLM2 <- lm(PREIS ~ MEILEN + SERVICE)
```

(lies: „lineares Modell von `PREIS` nach `MEILEN` und `SERVICE` (Anzahl der Serviceüberprüfungen)“). Dies ergibt

```
R> shcLM2

Call:
lm(formula = PREIS ~ MEILEN + SERVICE)

Coefficients:
(Intercept)       MEILEN      SERVICE
 6206.12836     -0.03146    135.83749
```

In diesem Modell reduziert sich also der Preis pro gefahrener Meile um USD 0.031; jedes zusätzliche Service erhöht den Preis um rund USD 136.

Mehr Information erhalten wir wieder mit

```
R> summary(shcLM2)

Call:
lm(formula = PREIS ~ MEILEN + SERVICE)

Residuals:
    Min      1Q  Median      3Q     Max
-86.079 -28.920   1.483  29.011  86.736

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.206e+03  2.497e+01  248.58   <2e-16 ***
MEILEN      -3.146e-02  6.319e-04  -49.79   <2e-16 ***
SERVICE      1.358e+02  3.903e+00   34.81   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 41.48 on 97 degrees of freedom
Multiple R-Squared: 0.9741,        Adjusted R-squared: 0.9735
F-statistic:  1822 on 2 and 97 DF,  p-value: < 2.2e-16
```

Durch Einführung der zweiten erklärenden Variablen `SERVICE` steigt das lineare Bestimmtheitsmaß auf $R^2 = 0.974$, es werden also rund 97 Prozent der Varianz von `PREIS` durch das Regressionsmodell erklärt.

5.5 Unterscheiden sich die Mittelwerte zweier Merkmale, die an der selben Beobachtungseinheit erhoben wurden?

Den „gepaarten“ t-Test (t-Test für abhängige Stichproben) führt man mit `t.test` aus, indem man als Argumente die Werte mit den Beobachtungen und zusätzlich `paired = TRUE` angibt (oder man bildet Differenzen und verwendet den Ein-Stichproben-Test).

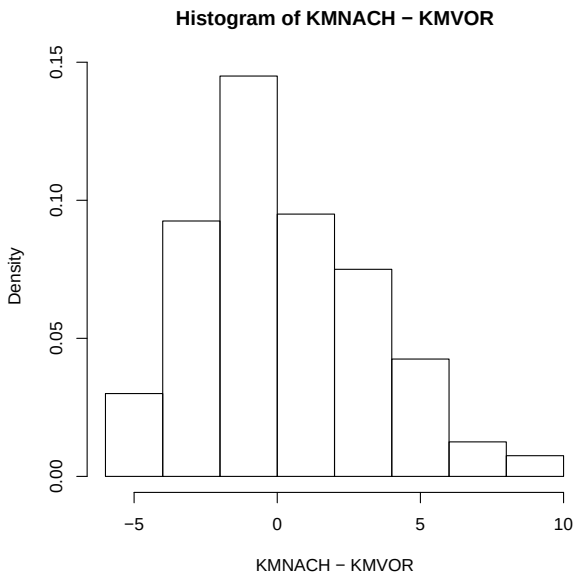
Beispiel 5.5 (Anzahl gefahrener Kilometer vor und nach Einführung der Helmtragepflicht)

Einlesen der Daten:

```
R> BICYCLE <- read.table("bicycle.tab", header = TRUE)
R> dim(BICYCLE)

[1] 200   2

R> names(BICYCLE)

[1] "KMNACH" "KMVOR"
```

Darstellung der Differenz der gefahrenen Kilometer nach und vor Einführung der Helmtragepflicht in einem Histogramm:

```
R> attach(BICYCLE)
R> hist(KMNACH - KMVOR, prob = TRUE)
```



Nach der Einführung scheint die Kilometerzahl sogar höher zu sein.

Um zu überprüfen, ob die Einführung eine *negative* Auswirkung hatte, müssen wir die Nullhypothese $\mu = 0$ gegen die Alternative $\mu < 0$ (Reduktion der Kilometerzahl) testen, wobei μ die Differenz der mittleren Kilometerzahlen in der Grundgesamtheit nach und vor Einführung der Helmtragepflicht ist:

```
R> t.test(KMNACH, KMVOR, paired = TRUE, alternative = "less")

        Paired t-test

data:  KMNACH and KMVOR
t = 3.5529, df = 199, p-value = 0.9998
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
     -Inf 1.128151
sample estimates:
mean of the differences
                   0.77
```

Der p-Wert ist de facto 1: die Einführung hat also „sicher“ nicht zu einer Reduktion geführt.

Zusammenräumen:

```
R> detach("BICYCLE")
R> remove(BICYCLE)
```

5.6 Unterscheidet sich die Lage zweier Merkmale, die an der selben Beobachtungseinheit erhoben wurden?

Den „gepaarten“ Wilcoxon-Test (Wilcoxon Matched Pairs Signed Ranks Test) führt man mit `wilcox.test` aus, indem man als Argumente die Werte mit den Beobachtungen und zusätzlich `paired = TRUE` angibt.

Beispiel 5.6 (Alkohol und Beurteilung der Attraktivität)

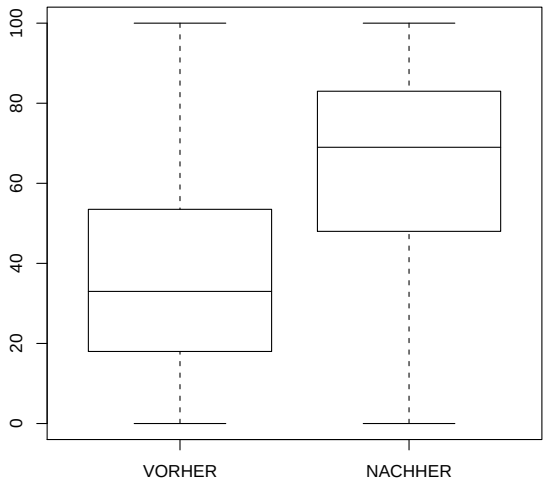
Einlesen der Daten:

```
R> ALCATTR <- read.table("alcattrneu.tab", header = TRUE)
R> dim(ALCATTR)

[1] 180   2

R> names(ALCATTR)
```

```
[1] "VORHER"  "NACHHER"
```

Für den grafischen Vergleich der Attraktivität verwenden wir nebeneinanderliegende Boxplots:

```
R> attach(ALCATTR)
R> boxplot(ALCATTR)
```



Die Attraktivität scheint also im Laufe des Abends deutlich gestiegen zu sein (ob aufgrund des Alkoholkonsums oder anderer Effekte soll dahingestellt bleiben).

Um zu überprüfen, ob die Einschätzung unmittelbar vor der Sperrstunde höher ist als 3 Stunden zuvor, müssen wir die Nullhypothese der Gleichheit der Lagen gegen die Alternative, dass der Median der Attraktivität „nachher“ größer ist als „vorher“, testen:

```
R> wilcox.test(NACHHER, VORHER, paired = TRUE,
+     alternative = "greater")

        Wilcoxon signed rank test with continuity
        correction

data:  NACHHER and VORHER
V = 13348, p-value = 5.551e-16
alternative hypothesis: true mu is greater than 0
```

Der p-Wert ist de facto 0; knapp vor der Sperrstunde werden Mitglieder des anderen Geschlechts also signifikant attraktiver eingeschätzt als drei Stunden davor.

Zusammenräumen:

```
R> detach("ALCATTR")
R> remove(ALCATTR)
```

Kapitel 6

Metrische und kategoriale Merkmale

6.1 Wie kann man metrische und kategoriale Merkmale numerisch beschreiben?

Typischerweise will man geeignete Maßzahlen (beispielsweise Lage- oder Streuungsmaße) für einzelne Gruppen getrennt berechnen und dann vergleichen. In R geht dies sehr einfach mit der Funktion `tapply`:

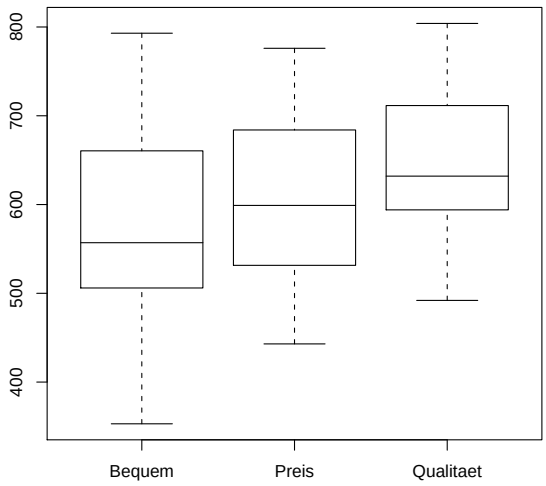
```
R> APPLE <- read.table("apple.tab", header = TRUE)
R> attach(APPLE)
R> tapply(VERKAUF, INHALT, mean)

   Bequem     Preis Qualitaet
   577.55    608.65    653.00

R> tapply(VERKAUF, INHALT, median)

   Bequem     Preis Qualitaet
      557       599       632

R> tapply(VERKAUF, INHALT, sd)

   Bequem     Preis Qualitaet
103.80268  93.11412  85.07705
```

berechnet Mittelwerte, Mediane und Standardabweichungen für die Verkaufszahlen getrennt nach Werbeinhalten.

6.2 Wie kann man metrische und kategoriale Merkmale grafisch beschreiben?

Zum Vergleich von Lage und Streuung in den einzelnen Gruppen verwendet man am einfachsten nebeneinanderliegende Boxplots:

```
R> boxplot(VERKAUF ~ INHALT)
```



(lies: „Boxplot von `VERKAUF` aufgeschlüsselt nach `INHALT`“).

6.3 Unterscheiden sich die Mittelwerte von 2 Gruppen?

Für den 2-Stichproben t-Test zum Vergleich der Mittelwerte („t-Test für unabhängige Stichproben“) verwendet man die Funktion `t.test`. Hat man zwei Vektoren von Beobachtungen für die jeweiligen Gruppen so nimmt man diese als Argumente; ansonsten, insbesondere wenn die Daten in einem data frame sind, verwendet man eine Modellformel in der die abhängige metrische Variable links und die erklärende kategoriale Variable rechts steht.

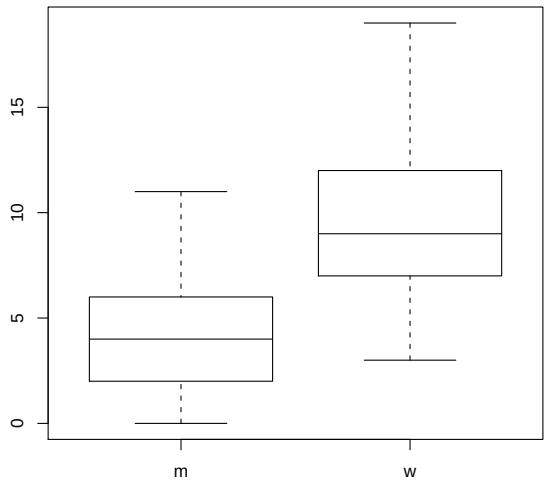
Beispiel 6.1 (Haushaltsarbeit von Teenagern)

Einlesen der Daten:

```
R> TEENAGEWORK <- read.table("teenagework.tab",
+     header = TRUE)
R> dim(TEENAGEWORK)

[1] 192   3

R> names(TEENAGEWORK)

[1] "STUNDEN" "MUTTER"  "SEX"
```

Wir haben also 192 Beobachtungen der 3 Variablen `STUNDEN`, `MUTTER` und `SEX`.

```
R> attach(TEENAGEWORK)
R> boxplot(STUNDEN ~ SEX)
```



Wir sehen, dass Mädchen also wesentlich mehr Stunden im Haushalt arbeiten als Burschen.

```
R> t.test(STUNDEN ~ SEX)

        Welch Two Sample t-test

data:  STUNDEN by SEX
t = -11.4432, df = 189.219, p-value = < 2.2e-16
```

```
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -5.776057 -4.077495
sample estimates:
mean in group m mean in group w
       4.439560        9.366337
```

Der im Boxplot erkannte Unterschied ist also auch im statistischen Sinne deutlich signifikant.

6.4 Unterscheidet sich die Lage eines Merkmals zwischen 2 Gruppen?

Für den 2-Stichproben Wilcoxon-Test zum Vergleich der Lage („Wilcoxon-Mann-Whitney-Test für unabhängige Stichproben“) verwendet man die Funktion `wilcox.test`. Hat man zwei Vektoren von Beobachtungen für die jeweiligen Gruppen so nimmt man diese als Argumente; ansonsten, insbesondere wenn die Daten in einem data frame sind, verwendet man eine Modellformel in der die abhängige metrische Variable links und die erklärende kategoriale Variable rechts steht.

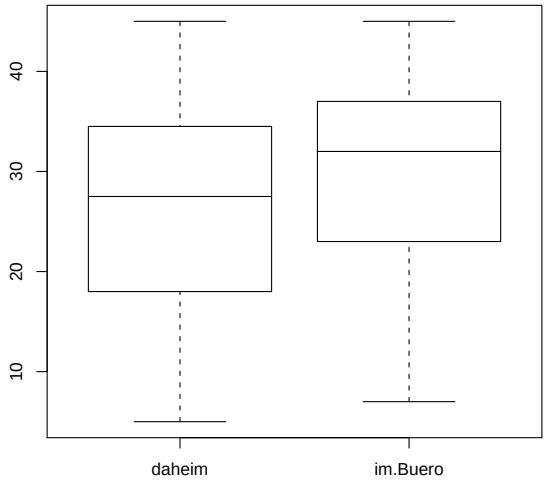
Beispiel 6.2 (Telearbeit und Arbeitszufriedenheit)

Einlesen der Daten:

```
R> TELEARBEIT <- read.table("comphomeneu.tab",
+     header = TRUE)
R> dim(TELEARBEIT)

[1] 200   2

R> names(TELEARBEIT)

[1] "ZUFRIEDENHEIT" "ORT"
```

Grafische Darstellung der Arbeitszufriedenheit in Abhängigkeit vom Arbeitsort:

```
R> attach(TELEARBEIT)
R> boxplot(ZUFRIEDENHEIT ~ ORT)
```



Interessanterweise scheint Heimarbeit eher unzufrieden zu machen (die erhöhte Flexibilität also weniger wichtig zu sein als die Reduktion sozialer Kontakte).

```
R> wilcox.test(ZUFRIEDENHEIT ~ ORT)

        Wilcoxon rank sum test with continuity
        correction

data:  ZUFRIEDENHEIT by ORT
W = 4319, p-value = 0.09614
alternative hypothesis: true mu is not equal to 0
```

Der Effekt ist aber nicht signifikant ($\alpha = 0.05$).

6.5 Unterscheiden sich die Mittelwerte mehrerer Gruppen?

Einfache Varianzanalyse kann man durchführen, indem man die Funktion `anova` auf ein angepasstes lineares Modell anwendet, wobei in der Modellformel die abhängige metrische Variable links und die erklärende kategoriale Variable rechts steht.

Beispiel 6.3 (Wirksamkeit verschiedener Werbeinhalte)

Einlesen der Daten:

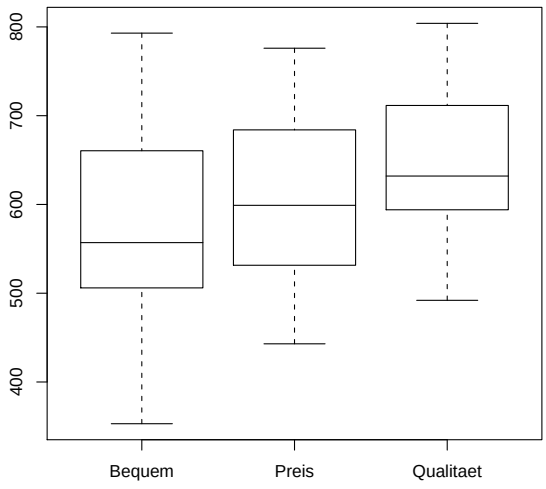
```
R> APPLE <- read.table("apple.tab", header = TRUE)
R> dim(APPLE)

[1] 60  2

R> names(APPLE)

[1] "VERKAUF" "INHALT"
```

Wir haben also 60 Beobachtungen von 2 Variablen, wobei

```
R> attach(APPLE)
R> table(INHALT)

INHALT
   Bequem     Preis Qualitaet
       20        20        20
```

in je 20 Fällen die Inhalte Preis, Qualität und Bequemlichkeit beworben wurden.
Im Boxplot

```
R> boxplot(VERKAUF ~ INHALT)
```



glauben wir einen deutlichen Vorteil des Werbeinhaltes „Qualität“ erkennen zu können; die einfache Varianzanalyse

```
R> anova(lm(VERKAUF ~ INHALT))

Analysis of Variance Table

Response: VERKAUF
          Df Sum Sq Mean Sq F value  Pr(>F)
INHALT     2  57512   28756   3.233 0.04677 *
Residuals 57 506983    8894
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

lässt die mittleren Verkaufszahlen als signifikant unterschiedlich zwischen den Inhalten erscheinen ($\alpha = 0.05$). (Beachte, dass dies nicht notwendigerweise die Hypothese ist, die wir prüfen wollen.)

(Der Vergleich von

```
R> tapply(VERKAUF, INHALT, mean)

   Bequem     Preis Qualitaet
   577.55    608.65    653.00
```

und

```
R> lm(VERKAUF ~ INHALT)

Call:
lm(formula = VERKAUF ~ INHALT)

Coefficients:
    (Intercept)      INHALTPreis  INHALTQualitaet
         577.55            31.10            75.45
```

zeigt, dass das lineare Modell genau die Unterschiede der Mittelwerte für die Inhalte Preis und Qualität zum Mittelwert für die „Basis“ (die erste der drei Kategorien) Bequemlichkeit erklärt.)

6.6 Unterscheidet sich die Lage eines Merkmals zwischen mehreren Gruppen?

Für die Kruskal-Wallis Varianzanalyse verwendet man die Funktion `kruskal.test` mit einer Modellformel, in der die abhängige metrische Variable links und die erklärende kategoriale Variable rechts steht.

Beispiel 6.4 (Fernsehkonsum von Kindern)

Einlesen der Daten.

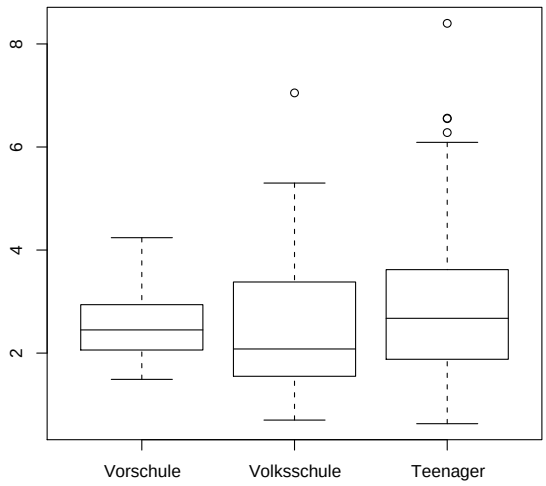
```
R> TV <- read.table("tv.tab", header = TRUE)
R> dim(TV)

[1] 150   2

R> names(TV)

[1] "STUNDEN" "GRUPPE"
```

Damit in der grafischen Darstellung die Altersgruppen in der richtigen Reihenfolge erscheinen, re-kodieren wir die kategoriale Variable `GRUPPE` als ordinal:

```
R> TV$GRUPPE <- ordered(TV$GRUPPE, levels = c("Vorschule",
+     "Volksschule", "Teenager"))
R> attach(TV)
```

Grafische Darstellung durch nebeneinanderliegende Boxplots:

```
R> boxplot(STUNDEN ~ GRUPPE)
```



Die vielleicht vermutete ausgeprägte Zunahme des TV-Konsums in Stunden mit dem Alter ist nicht zu erkennen. Auch der Kruskal-Wallis Test

```
R> kruskal.test(STUNDEN ~ GRUPPE)

        Kruskal-Wallis rank sum test

data:  STUNDEN by GRUPPE
Kruskal-Wallis chi-squared = 2.3113, df = 2,
p-value = 0.3148
```

zeigt, dass die Nullhypothese der Gleichheit der Lage nicht verworfen werden kann. (Wir beobachten allerdings, dass die Streuung mit dem Alter zu wachsen scheint.)

Index